Identity-as-a-Service (IDaaS): a Missing Gap for Moving Enterprise Applications in Inter-Cloud

Tri Hoang Vo
Deutsche Telekom
Berlin, Germany
Tri.Vo-Hoang@telekom.de

Woldemar Fuhrmann
University of Applied Sciences Darmstadt
Darmstadt, Germany
w.fuhrmann@fbi.h-da.de

Klaus-Peter Fischer-Hellmann
Digamma GmbH
Darmstadt, Germany
K.P.Fischer-Hellmann@digamma.de

Abstract—Migration of existing enterprise applications to the Cloud requires heavy adaptation effort in individual architectural components of the applications. Existing work has focused on migrating the whole application or a particular component to the Cloud with functional and non-functional aspects. However, none of them has focused so far on the adaptation of web service security. Towards this goal, we focus on the adaptation of web service security for migrating applications from local hosting to the Cloud, and for moving applications in Inter-Cloud environment. Identity-as-a-service (IDaaS) decouples web service security from the business logic as a manageable resource during the life cycle of an application in the Cloud environment. On the other hand, IDaaS provides identity roaming for Cloud users to access multiple service providers on demand, but also preserve user's privacy. IDaaS coordinates automated trust negotiation between Cloud users, who want to enforce their data privacy, and service providers, who have heterogeneous security policy in federated security domains. In this paper, we first introduce IDaaS with scenarios and new requirements in comparison to traditional Identity Management systems, and propose a brief model for IDaaS.

Keywords—identity as a service; federated identity management; inter-cloud; identity roaming; attribute-based access control; privacy-aware access control

I. Introduction

In a local hosting environment, traditional applications have their own implementation for authentication and authorization. One local application has *n* user accounts. The management process of user identities associated with different levels of access control gave birth to Identity Management (IDM). In Cloud computing, enterprise applications come from federated security domains; provide themselves as a Service Provider (SP) and cooperate with each other. From the beginning, they might adapt their local IDM with one another manually. However as time goes by, their applications migrate to another Cloud provider, or their partner service de-provisioning from the current Cloud provider, so they need to synchronize and grant new access controls upon each change. In such a dynamic provisioning environment on the Cloud, enterprise applications may prefer outsourcing their security implementation to a third party central service [1]. In this case, their security can be strengthened by a specialized provider and reduce their cost.

A traditional authorization system is Role Based Access Control (RBAC), whereby access control depends on the role of an entity after it authenticates to the system. Developers of the application also tend to hardcoding the authorization logics into the source code of the program and only these developers can understand it. Authorization hardcoding is in contrast to a policy-based approach, in which administrators can manage different rules and dynamically changed interpreted logic without modifying the underlying implementation [2]. Without policy-based approach, it is a big obstacle for applications to be portable on the Clouds, because developers have to adapt source code to various security domains with different security policies.

The Attribute Based Access Control (ABAC) is a policy-based approach for fine-grained authorization because access control in this approach does not require identifying the entity as a whole, but depends on its attributes, for example: "A user can buy a specific DVD if he is an adult". XACML (eXtensible Access Control Markup Language) [3] is a standard policy language and supports ABAC. A reference architecture includes four main components: Policy Decision Point (PDP) evaluates access request against the security policies, Policy Enforcement Points (PEP) enforces authorization decision from PDP, Policy Information Point (PIP) stores and collects missing user attributes in the request, and Policy Administration Point (PAP) administrates the policy. Gartner predicts: "By 2020, 70% of all businesses will use ABAC as the dominant mechanism to protect critical assets" [4]. This is the reason why we focus on IDM not only for the traditional RBAC, but also for ABAC. Existing work adapted functional components of applications to the Cloud environment [5]. The adaptation for authentication and authorization has security challenges and needs more research.

From the user's perspective, users may have multiple identities with public and private profiles in the Internet. They may prefer to access multiple SPs in federated security domains, but also preserve their privacy. When users do not store their personal identities in a local machine, but in an SP in the Cloud, they may be interested in questions such as where their data are stored actually, how secure it is, who can access it except themselves. Even if SPs specified their privacy policies, we cannot guarantee that they will follow their policies and will not transfer user data to another party. If users cannot trust any third parties to hold their data, they might prefer to trust themselves. In a user-centric approach, users actively decide which identity credentials they want to exchange with which SPs. They can selectively disclose a minimal attribute set [6] or hide their identities by using anonymous credentials in transactions with SPs [7] [8].

However, user-centric approach has a limitation for the users themselves because it overloads the IDM tasks to the user's decision in every transaction. Moreover, anonymous credentials require SPs to accept it and to develop an adaptation. But most SPs, who select an Identity Provider (IdP) based on how much information they can collect from the user [9], are not likely to support anonymous credentials.

OASIS Identity in the Cloud TC [10] mentioned Identity-as-a-Service (IDaaS) as an approach to identity management in which an entity (individual or organization) relies on special service provider's functionalities that allows the entity to perform an electronic transaction, which requires identity data managed by this provider. Since the definition is non-standard and coarse, in the following paper we revisit the requirements of a traditional IDM system by Kim Cameron [11] and specify which requirements are still missing in Cloud computing. An IDM system is much likely to succeed when it benefits to all parties, including users, SPs and IdP [9]. This is our goal to propose such a model for IDaaS.

In the first place, we will start with scenarios and analyze missing requirements of IDaaS in section II and III, respectively. In section IV, we will propose a draft model of IDaaS: how it should look, and whether the given model could be successful, based on lessons learned from failures of IDM system in the past. In the end, we will summarize related work that helped us to understand and to propose the draft model.

II. Motivating scenarios

IBM's Cloud Computing Reference Architecture [12] defines three main roles in any cloud computing environment: a service creator develops a service running on one or more platforms, a Cloud provider who runs those services on demand, and Cloud consumers or end users who consume these services. Based on these roles, we define the following scenarios. A Cloud provider, for example Amazon WS, opens a Software (or Platform) as a service business market place. Service creators register two SPs in the repository: An office service for multiple users to corporate their daily work and a storage service for saving data. The following issues come up very often:

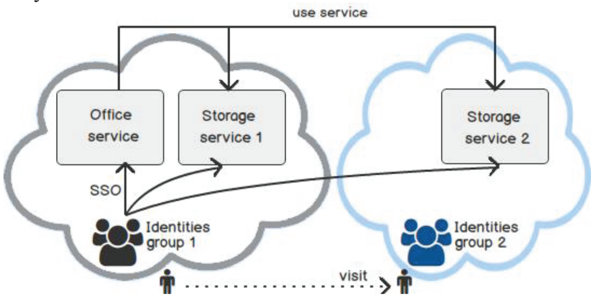

Fig. 1. User visits another Cloud in distant location

1) Dynamic Single-Sign-On (SSO): The Cloud provider already has a number of users, who agreed on a set of policies for data privacy. They signed the contracts and provided their billing information. Fig. 1 shows an end user who uses the two Cloud services. Without Single-Sign-On support, he has to use two different credentials: one credential to download his data from the storage service locally and another one to re-upload the data to the office service, although the two services reside on the same Cloud provider. On the other hand, the office service comes from a local environment. It has its own user accounts and security policies. The office service might offer its local users to SSO to other Cloud services. In traditional IDM, tenant administrator pre-configures IdP manually as trust parties to verify user's authentication request and issue authentication token for users to SSO to multiple relying services. Problems may occur when privacy policies of Cloud users and the security policies of relying services conflict with each other.

2) Dynamic service binding: The Cloud provider may have a statistic report about the business market place. The report may indicate that most users, who use the office service, also use the storage service. Based on such a statistic report, the creator of the office service might want to cooperate with the storage service as its persistent backend. The current solution in IDM provides an IdP to issue credentials for each relying services to trust and authorize access control. However, relying services have to configure their implementation manually. As an independent software vendor (ISV), the service creator may have designed his application to run on multiple platforms and may refuse fixed bindings to any web services. The statistic report is only true in the current business marketplace, but in another market, the office service might cooperate with a different storage service.

3) Identities roaming: A user uses an interactive application like online playing game and moves to geographically distant locations (another city, country or continent) temporally. Due to long distance implying many intermediate nodes at different sites, a query transaction to the database may take 200 ms or even longer to complete. Therefore, access control may have a downgrade in performance if the PDP and PIP of his home provider are far from the PEP of the application. In best practice, mobile users authenticate to an endpoint closest to their location and exchange a symmetric key before they may access to further resources in the data center of an application [10]. As a result, user attributes may be temporally migrated to the visitor provider closest to the user location. The visitor Cloud provider may host another instance of the same distributed application but the user has no billing contracts with this provider. As a citizen who works for a government in Europe, for instance, he may not disclose his personal information in country C according to the EU Data Protection Directive [13]. This scenario raises a question how we can protect user privacy but satisfy access control to an SP, which belongs to different federated security domains.

III. IDaaS Requirements

By learning from the failure of IDM systems like Microsoft's Passport in the past, Kim Cameron pointed out "*the seven laws of Identity*" to build a successful IDM system [11]. Briefly, an IDM system has to reveal least user information with the user's consent to limited parties, which necessarily participate in the transaction. It has to support both private and public profiles for an entity. It should not limit users to only one Identity Provider, but support interoperability between them so that users can easily switch between pseudonymous identities, and provides users with a simple, consistent experience through multiple technologies. Our scenarios given in section II emphasized that Cloud is a dynamic environment, whereby applications provision and de-provision frequently; users change their locations and access multiple SPs from federated security domains. Therefore, we should extend the requirements of traditional IDM as follows:

1) Control the life cycle of Authentication and Authorization Infrastructure (AAI): Since the SPs may hesitate to change their implementations in order to adapt to diverse hosting environments, IDaaS can support the adaptation of SPs by defining service components for authentication and authorization. It decouples AAI from the business logic of the application. The security service components should have well-defined configurable templates with interfaces and parameters like with the concept of Service Component Architecture (SCA) [14] such that values can be entered at the time the service is provisioned to the Cloud. The lifecycle of such security components should be well defined and orchestrated automatically together with the lifecycle of the Cloud application.

2) Scalability: A new deployed application should not affect any running applications already existing on the platform; otherwise, any changes would require a huge cascading refresh of all tenants on the platform.

3) Automated trust negotiation: Traditional IDM system requires administrators to pre-configure the IdP to trust the authority of a partner service manually. IDaaS can support handshakes between parties so that no information is revealed, until the negotiation process completes and all parties can start their business transactions.

4) Privacy protection for identity roaming: IDaaS can support users in the identity-roaming scenario by protecting their identities with privacy-aware access control. Traditional IDM is missing a mechanism to protect user data when it is temporally roaming to a new location.

5) Performance: XACML has a limitation in performance [15]. For each user request to a resource, PEP has to interact with its PDP to ask for authorization decision, and PDP further queries user attributes from PIP for taking decision. If user attributes are stored in external location or distributed in federated domains, PIP has latency to collect and synchronize user attributes. As a result, authorization decision has a delay for each user request to a resource, or fails to take decision when external resource is not available. Therefore, we should keep user identities (in PIP), access control policies (in PDP), and policy enforcement point (PEP) for an authorization request of an application closed to each other. An Identity Server typically enables caching for policies and user attributes, and also enables user provisioning from external resource to local IDM system [16].

The first two requirements support the adaptation of application to the Cloud for *dynamic SSO*, *service-binding* scenarios. The later three requirements support *identity roaming in federated domains* scenarios.

IV. IDAAS PROPOSED MODEL

A. Trust model

The dynamic provisioning of SPs on a Cloud platform requires dynamic trust establishments between several parties: Trusts between Cloud users and SPs in the same or in different Cloud providers. Trust establishment should take advantages of exiting trust relationships. For instance, users and SPs have signed contracts with their Cloud provider, so they built bilateral direct trusts: users trust Cloud provider to administrate and provide lawful SPs and protect their data according to the user's privacy policy description, SPs trust their Cloud provider to provide natural users and accountable for their actions with a billing system. Fig. 2 shows a home provider with bilateral trust and a visitor provider with which users and SPs have no contracts.

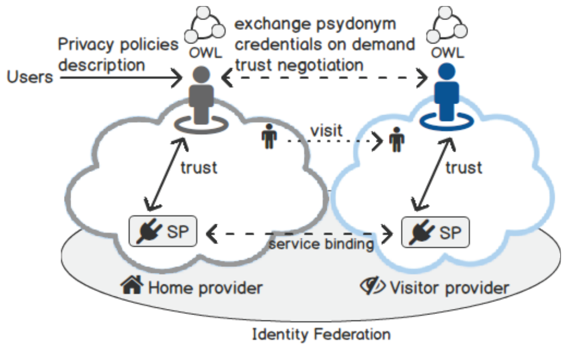

Fig. 2. IDaaS trust model

From the user's perspective, IDaaS of the home provider (home IDaaS) should be a central place to collect all user attributes (due to the bilateral trust). It includes user basic information when the users registered on the business market place (e.g., name, age), billing information (e.g., PayPal account), and contact information (e.g., mobile number). The home IDaaS can also collect positive attributes from one SP (according to the user's privacy policies), for instances reviews and ratings. If user data are distributed within all service members and are collected just in time for access control, the performance may degrade depending on the data distribution grades in multiple external resources. Moreover, if the users have no direct contracts with the SPs, the SPs have no right to collect all user attributes.

From the perspective of SPs, home IDaaS is the only party that SPs can trust and contact for handling requests. Automated trust negotiation between users and an SP in the same local domain as well as between federated domains is the responsibility of home IDaaS and not of the SPs themselves. SPs should only concentrate on developing and providing their business services.

B. *IDaaS components*

Fig. 3 shows several components of the IDaaS with separation of duty. We reuse the reference architecture XACML and add additional functionalities to satisfy our requirements in section III:

1) Policy Enforcement Point: This plugin intercepts authentication request and handle authorization for the SP. When the SPs are provisioned to a Cloud platform, PEP is a configurable module depending on adaptive information from a deployment process of the Cloud provider's orchestration engine.

2) Policy Decision Point: Ideally, IDaaS can support SP's designers by giving them an inference mechanism to analyze and derive any elements related to Security Policy from an existing implementation during the pre-configuration phase of the service. Security policy should be derived semi- or fully-automatically and published in a uniform way (by using XACML privacy policy profile [17]) to a central service of the Cloud provider to facilitate automated trust negotiation with other partner services in the Cloud.

3) Policy Information Point: PIP in the reference architecture of XACML provides user information for PDP to make decisions. In IDaaS, this component also handles identity roaming between IDaaS in different security domains. Fig. 3 shows that PIP can be an external service, whereby a set of users signed the contract with, but independently from the Cloud provider.

4) Policy Administration Point: Is an endpoint (not shown in Fig. 3) to provide functionalities for operators of tenant application to review the derived policies in (2) and configure on demand.

5) Orchestration engine: Is the central service of a home Cloud provider, to orchestrate the life cycle of AAI in the provider (described next in section C).

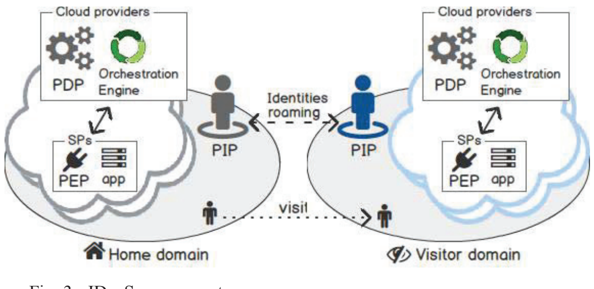

Fig. 3. IDaaS components

C. *Control the life cycle of AAI*

Enterprise Architecture Management (EAM) defines the guidelines, design principles, and evolution paths for enterprise IT [18]. However, no generally accepted model to denote enterprise architectures has evolved yet. Most enterprise applications have no clear picture about their topology. The more complex the system, the more effort and error-proneness is involved in adaptation upon topology changes [5].

Topology and Orchestration Specification for Cloud Applications (TOSCA) [19] is recently well-known as the upcoming standard to describe topology for Cloud applications [20]. TOSCA describes each component in the application by deriving from a *node type* (with specific *properties* and *interfaces*). Components connect to each other through *relationships* (with *requirements*, and *capabilities*). A *plan* describes the management lifecycle of service creation and termination. When developers describe an application's topology in TOSCA, they can deploy the application to any platforms that support a TOSCA orchestration engine.

The concept of AAI separates the security implementation from the web services so developers can focus on implementation of business logic only. So far, TOSCA does not have elements to describe the notation for WS-Security component. XACML does not have elements to describe the parameters and interfaces of a PEP. Therefore, we will consider extending (or inheriting) TOSCA elements for describing the PEP, PDP and PIP modules. For instance, PEP can have special *properties*, (e.g., public/private keys, identity service endpoint) and *interfaces*, (e.g., interface to update service policy and publish local user attributes in the scope of the application, or interface to retrieve updates from a deployment process to establish trust with an entity). In short, a Cloud provider orchestration engine can setup, update and clean up configurable WS-Security components during the life cycle of an application until the application terminates its contract to a Cloud provider.

D. *Identity Roaming*

OECD stated eight privacy guidelines that every IDM should follow [21]. Briefly, the collection of personal data should be lawful under the consent of users and with a specified purpose for which they are used. The purpose for data collection has a time limit (from the time to collect data to the time to fulfill the purpose). For later use, personal data should not be disclosed for other purposes than the ones they have been collected for, as well as are to be protected from unauthorized access and modification. In addition, users have the right to control their data.

User identities are roaming to another party for the purpose of authentication and authorization to an SP in a visitor domain. We consider these guidelines in a visitor domain as follows. Due to the "*use limitation principle*", the roaming data should not be visible to any entity except for the PDP in the visitor IDaaS, for which purpose the data is collected. For instance, system administrators cannot read roaming data from the database in plain text. After the purpose is fulfilled, due to the "*purpose specification principle*", the roaming data should be deleted or self-destroyed.

Moreover, in our proposed model, only the home IDaaS knows the real user behind the transaction for billing purposes. The visitor IDaaS as well as the SP in the visitor domains will only know a pseudonym userid and the identifier of the home IDaaS where the pseudonym was issued, so they can send requests for payment to the home IDaaS. This is particularly similar to roaming mobile networks where the visitor network contains neither all user information of the visiting mobile in his network nor the secret authentication key to identify this mobile.

In comparison to the pseudonym system *Idemix* [7], we have a more relaxed level of protection because it cannot protect user privacy when the home and the visitor IDaaS, or the home IDaaS and an SP cooperate with each other to reveal the user identity. Here the home IDaaS may play the roles as credential issuer, revocation authority, and credential inspector when something goes wrong. The computation resource is moved from the user client to a high performance server in his home IDaaS.

However, the OECD privacy guidelines cannot protect user data from a system that steals personal data unlawfully (by programmatically implementing an interceptor to dump unencrypted data from the memory of the authorization service and transfer it to a third party).

V. Related Work & discussion

In the following section we first summarize design patterns of AAI collected from [22] [23] and point out which pattern fits our needs.

1) Isolated IDM: Because a third party can be an untrusted host and target to correlation attacks, this design does not reply on a Trusted Third Party (TTP) for issuing and verifying credentials. Work in [24] gives user the possibilities to issue his encrypted credentials and give them to SPs. The SPs can then evaluate the predicate of the given encrypted credentials by contacting multi-party computing. It does not mention how the users can prove that he really possesses his attributes, because without a TTP, the user can only self-assert his attributes. The major limitation of such approach is scalability when the number of users and SPs increase.

2) Circle of Trust (CoT): This pattern requires all SPs completely trusting each other to exchange information about their subjects in operational agreements and to exchange data in business agreements. As a result, users may authenticate to an IdP of one SP and consume all other services in this CoT. However, user attributes are exchanged between all services in a domain of trust with no filtering and protections. An existing implementation of this pattern is Liberty ID-FF [25].

3) Centralized IDM: In contrast to the CoT, this pattern centralizes the administration of identity. A central IdP is responsible for storing and providing identity to SPs within a security domain. Microsoft Passport is an example of this pattern, which failed in the past because it violates the rule justifiable parties of the "seven laws of Identity" [26].

4) Identity Federation: This pattern concerns identities across multiple security domains. A unique IdP manages local identities in one security domain and a user may have multiple local identities with some SPs. A SP from domain 1 forms a federation with a SP from domain 2 by developing offline-operating agreement. Identity of a user may have various presentations in different domains and a federated identity may be gathered from all domains to consolidate in one place. Either a central service collects federated attributed about a user from all federated members as in [1] [27], or an SP actively queries local identities and transfers all attributes from domain 1 to his peer service in domain 2 as in [22].

GEYSERS, an EU project described in [28], allows e-research users and groups to provision dynamic infrastructure in IaaS provider. It uses Enterprise Service Bus (ESB) and WS-Trust to provide trust binding between services. This solution perfectly fits in academic research projects and Federated Cloud when services volunteer to use the same framework implementation. Notice that we do not consider ESB a solution for IDaaS, because there are no global standard concepts or implementations for ESB. ESB vendors define their own messaging encapsulation differently and it is not mandatory for all IDaaS providers to use the same vendor implementation. Our focus is Inter-Cloud computing targeting general Cloud services that straightforward interact with each other with no tight binding in vendor implementation.

5) User-centric: This IDM involves users as a man in the middle to retrieve credentials from an IdP and selectively decide which credentials to exchange to SPs. Thus, the users avoid direct contact between Identity Provider and SPs. Idemix [29] [30] [7] introduced an approach for a user to hide his identity by using pseudonym and anonymous credentials in transactions with SPs. User first obtains credentials from an issuing organization. Based on the issued credential he can use his secret master key to generate any pseudonyms and subset of credentials to show in following transactions with SPs that trust the issuing organization. The issuing organization does not need to be available during the transactions between users and SPs. Even if the issuing organization and the SPs cooperate with each other, they cannot link all transactions together to reveal the identity of the user. Based on Idemix, ABC4Trust [31] provided a language framework for attributes based credentials. It defines an XML schema to present pseudonym, anonymous credentials, service policy, and demonstrate an integration with standard protocols like WS-Trust, SAML [32].

Discussion: Using Inter-Cloud taxonomy [33], we think *Federated Cloud* fits the CoT pattern because Cloud providers volunteer to share their resources and academic research projects may get the most benefits from it. On the other hand, *Multi-Cloud* fits *Identity Federation* because this concerns multiple security domains of independent vendors and demands transfer of attributes between domains. A *User-centric* approach, however, delegates the Identity Management tasks to the users. The "*seven laws of Identity*" stated that human and PC is the weakest link compared to the links between PC, SPs and Identity Providers. Therefore, the human link is the major target of Identity theft [26]. ABC4Trust might need to outsource some management tasks to IDaaS, but still provides the basic features such as selective disclosure of attributes and pseudonym. ABC4Trust may also lack support from SPs. In this case, an orchestration engine of IDaaS in our research to adapt SPs to the Cloud may be very helpful. None of the existing work supports an orchestration engine to adapt and control the AAI of SPs to an existing Cloud platform, the privacy protection for identity roaming, as well as automated trust negotiation.

VI. Summary and Future Work

In this paper, we first collected scenarios, new requirements of IDaaS in addition to a traditional IDM and proposed a draft model for IDaaS. We mention our future works as follows:

1) We will extend TOSCA specification language to describe a model for IDaaS components (PIP, PDP and PEP) as plugin modules of Cloud applications.

2) According to the privacy guidelines principles, we may develop a mechanism to protect the identity roaming against identity theft by making them available to the authorization service only on user demand.

The separation of PIP as an external service has a benefit to take advantages of existing identity providers to establish trust between users and Cloud providers. The most attractive identity providers are mobile network operators, since they already have mobile users, who are willing to access multiple SPs on a Cloud provider. These operators also have signing contracts for mobile roaming with each other, so they can easily adapt to IDaaS for identity roaming and billing system. In future work, we will consider automated trust negotiation between IDaaS based on existing trust between mobile network operators.

References

[1] C. Schläger, M. Sojer, B. Muschall, and G. Pernul, "Attribute-Based Authentication and Authorisation Infrastructures for E-Commerce Providers," in *E-Commerce and Web Technologies SE - 14*, vol. 4082, K. Bauknecht, B. Pröll, and H. Werthner, Eds. Springer Berlin Heidelberg, 2006, pp. 132–141.

[2] Wikipedia contributors, "Policy-based management," *Wikipedia, The Free Encyclopedia*, 2016. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Policy-based_management&oldid=700392159. [Accessed: 18-Jan-2016].

[3] "eXtensible Access Control Markup Language (XACML) Version 3.0," *OASIS Standard*, 2013. [Online]. Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html.

[4] R. Wagner, "Identity and Access Management 2020," *ISSA J.*, 2014.

[5] V. Andrikopoulos, T. Binz, F. Leymann, and S. Strauch, "How to adapt applications for the Cloud environment," *Computing*, vol. 95, no. 6, pp. 493–535, Dec. 2012.

[6] A. Nanda and M. B. Jones, "Identity selector interoperability profile v1. 5," *Microsoft Corporation*, 2008. [Online]. Available: http://download.microsoft.com/download/1/1/a/11ac6505-e4c0-.

[7] J. Camenisch and E. Van Herreweghen, "Design and implementation of the idemix anonymous credential system," *9th ACM Conf. Comput. Commun. Secur. CCS 02*, p. 21, 2002.

[8] D. B. Skillicorn and M. Hussain, "Personas : Beyond Identity Protection by Information Control A Report to the Privacy Commissioner of Canada," *Identities*, no. April, 2009.

[9] S. Landau and T. Moore, "Economic tussles in federated identity management," *First Monday*, vol. 17, no. 10, 2012.

[10] "Identity in the Cloud Use Cases Version 1.0," *OASIS Committee Note 01*, 2012. [Online]. Available: http://docs.oasis-open.org/id-cloud/IDCloud-usecases/v1.0/cn01/IDCloud-usecases-v1.0-cn01.html.

[11] K. Cameron, "The Laws of Identity," 2005. [Online]. Available: http://www.identityblog.com/?p=352/#lawsofiden_topic3.

[12] M. Behrendt, "IBM Cloud Computing Reference Architecture," *Open Group submission*, 2011. [Online]. Available: http://www.opengroup.org/cloudcomputing/uploads/40/23840/CCRA.IBMSubmission.02282011.doc.

[13] F. H. Cate, "The EU Data Protection Directive, Information Privacy, and the Public Interest," *80 Iowa Law Rev. 431*, p. 646, 1995.

[14] M. Edwards, "Service Component Architecture Specification (SCA)," *OASIS Standard*, 2007. [Online]. Available: http://www.oasis-opencsa.org/sca.

[15] G. Gebel, "Authorization Performance Myth Busting," 2010. [Online]. Available: http://analyzingidentity.com/2010/04/30/authorization-performance-myth-busting/. [Accessed: 30-Apr-2010].

[16] S. Gnaniah, "Improving XACML PDP Performance with Caching Techniques," *WSO2 Identity Server 5.1.0 Documentation*, 2015. [Online]. Available: http://docs.wso2.com/.

[17] "XACML v3.0 Privacy Policy Profile Version 1.0.," *OASIS Committee Specification 02*, 2015. [Online]. Available: http://docs.oasis-open.org/xacml/3.0/privacy/v1.0/xacml-3.0-privacy-v1.0.html.

[18] R. Winter and R. Fischer, "Essential Layers, Artifacts, and Dependencies of Enterprise Architecture," in *2006 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06)*, 2006, pp. 30–30.

[19] "Topology and Orchestration Specification for Cloud Applications Version 1.0," *OASIS Standard*, 2013. [Online]. Available: http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html.

[20] T. Binz, U. Breitenbücher, O. Kopp, and F. Leymann, "TOSCA: Portable Automated Deployment and Management of Cloud Applications," in *Advanced Web Services SE - 22*, A. Bouguettaya, Q. Z. Sheng, and F. Daniel, Eds. Springer New York, 2014, pp. 527–549.

[21] "Guidelines on the protection of privacy and transborder flows of personal data," *OECD*, 1980. [Online]. Available: http://www.oecd.org/document/18/0,3343,en_2649_34255_1815186_1_1_1_1,00.html.

[22] N. Delessy, E. B. Fernandez, and M. M. Larrondo-Petrie, "A Pattern Language for Identity Management," *Comput. Glob. Inf. Technol. 2007. ICCGI 2007. Int. Multi-Conference*, pp. 31–31, 2007.

[23] U. Habiba, R. Masood, M. Shibli, and M. Niazi, "Cloud identity management security issues & solutions: a taxonomy," *Complex Adapt. Syst. Model.*, vol. 2, no. 1, pp. 1–37, 2014.

[24] R. Ranchal, B. Bhargava, L. Ben Othmane, L. Lilien, A. Kim, M. Kang, and M. Linderman, "Protection of Identity Information in Cloud Computing without Trusted Third Party," in *2010 29th IEEE Symposium on Reliable Distributed Systems*, 2010, pp. 368–372.

[25] "Liberty Alliance project." [Online]. Available: http://www.projectliberty.org.

[26] D. Chadwick, "Federated Identity Management," in *Foundations of Security Analysis and Design V SE - 3*, vol. 5705, A. Aldini, G. Barthe, and R. Gorrieri, Eds. Springer Berlin Heidelberg, 2009, pp. 96–120.

[27] C. Schläger, T. Nowey, and J. a. Montenegro, "A reference model for authentication and authorisation infrastructures respecting privacy and flexibility in b2c eCommerce," *Proc. - First Int. Conf. Availability, Reliab. Secur. ARES 2006*, vol. 2006, pp. 709–716, 2006.

[28] Y. Demchenko, C. Ngo, C. de Laat, D. Lopez, A. Morales, and J. García-Espín, "Security Infrastructure for Dynamically Provisioned Cloud Infrastructure Services," in *Privacy and Security for Cloud Computing SE - 5*, S. Pearson and G. Yee, Eds. Springer London, 2013, pp. 167–210.

[29] D. Chaum, "Security without identification: transaction systems to make big brother obsolete," *Commun. ACM*, vol. 28, no. 10, pp. 1030–1044, 1985.

[30] J. Camenisch and A. Lysyanskaya, "An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation.," *Adv. Cryptol. - EUROCRYPT 2001, Int. Conf. Theory Appl. Cryptogr. Tech. Innsbruck, Austria, May 6-10, 2001*, vol. 2045, pp. 93–118, 2001.

[31] K. Rannenberg, J. Camenisch, and S. Ahmad, *Attribute-based Credentials for Trust*. Springer, 2015.

[32] S. Cantor and I. Kemp, "Assertions and protocols for the oasis security assertion markup language," *OASIS Stand. (March …*, no. March, pp. 1–86, 2005.

[33] N. Grozev and R. Buyya, "Inter-Cloud architectures and application brokering: Taxonomy and survey," *Softw. - Pract. Exp.*, vol. 44, no. 3, pp. 369–390, 2014.