Abstract
The emergence of cloud computing paradigm offers attractive and innovative computing services through resource pooling and virtualization techniques. Cloud providers deliver various types of computing services to customers according to a pay-per-use economic model. However, this technology shift introduces a new concern for enterprises and businesses regarding their privacy and security. Security as a Service is a new cloud service model for the security enhancement of a cloud environment. This is a way of centralizing security solutions under the control of professional security specialists. Identity and access control services are one of the areas of cloud security services, and sometimes, are presented under the term Identity as a Service.

This master thesis research is focused on identity-security solutions for cloud environments. More specifically, architecture of a cloud security system is designed and proposed for providing two identity services for cloud-based systems: authentication and authorization. The main contribution of this research is to design these services using service-oriented architectural approach, which will enable cloud-based application service providers to manage their online businesses in an open, flexible, interoperable and secure environment.

First, the architecture of the proposed services is described. Through this architecture all system entities that are necessary for managing and providing those identity services are defined. Then, the design and specification of each service is described and explained. These services are based on existing and standardized security mechanisms and frameworks. As a demonstration, a prototype system of an authorization service is implemented and tested based on the designed authorization solution. The implementation is done using Web Service technology respective to the service-oriented design approach. It is shown that both services are at least computationally secure against potential security risks associated with replay attacks, message information disclosure, message tampering, repudiation and impersonation.

The designed security system ensures a secure and reliable environment for cloud-based application services which is very easy to deploy and exploit on cloud-based platforms.
To my beloved parents:

Norayr Hakobyan

Hasmik Hovhannisyan
Acknowledgements

I am honored to work with my supervisor, Professor Sead Muftic, and I would like to thank him for his support and patient guidance during my master thesis work.

I am also thankful to the whole SecLab team for a helpful collaboration.

Finally, I would like to thank my whole family for their encouragement and invaluable support during this work period.
## Contents

Abstract ........................................................................................................................................... 3  
Acknowledgements .......................................................................................................................... 5  
List of Figures .................................................................................................................................. 8  
List of Tables ..................................................................................................................................... 9  

1. Introduction .................................................................................................................................... 10  
   1.1. Background .............................................................................................................................. 10  
   1.2. Problem Statement ................................................................................................................... 12  
   1.3. Purpose and Goals .................................................................................................................... 13  
   1.4. Research Methodology ........................................................................................................... 13  
   1.5. Thesis Outline ......................................................................................................................... 13  

2. Related Work ................................................................................................................................... 15  
   2.1. Overview of Secure Identity Management Systems .................................................................. 15  
      2.1.1. Identity Ecosystem ........................................................................................................... 15  
      2.1.2. ICAM Identity Authentication .......................................................................................... 16  
      2.1.3. Cloud SSO Authentication .............................................................................................. 18  
   2.2. Overview of Service-Oriented Architecture ............................................................................. 20  
   2.3. Web Service Security Considerations ...................................................................................... 21  
   2.4. Overview of the SAML Standard ............................................................................................. 23  
      2.4.1. SAML Concepts and Components ................................................................................... 24  
      2.4.2. SAML Privacy and Security Features .............................................................................. 24  
   2.5. Overview of the XACML Standard ........................................................................................... 25  
      2.5.1. XACML Components ....................................................................................................... 26  
      2.5.2. XACML Security and Privacy Considerations ................................................................... 27  

   3.1. Overview of Cloud Architecture Model ................................................................................... 28  
   3.2. Security System Design Approach ......................................................................................... 29  
   3.3. Overview of the System Entities .............................................................................................. 29  
      3.3.1. Authentication System ....................................................................................................... 31  
      3.3.2. Authentication System Security Protection ......................................................................... 33
List of Figures

Figure 1: RP-initiated Use Case [18] ................................................................................................................. 17
Figure 2: Trust Model [18] ...................................................................................................................................... 18
Figure 3: Sequence Diagram of the CCAA-SSO profile Authentication [19] .......................................................... 19
Figure 4: WS participants ........................................................................................................................................ 20
Figure 5: SAML Components [35] .......................................................................................................................... 24
Figure 6: XACML Context [40] .............................................................................................................................. 26
Figure 7: Cloud Architecture Model ....................................................................................................................... 28
Figure 8: Central Security System ........................................................................................................................... 30
Figure 9: Cloud Security Infrastructure .................................................................................................................. 31
Figure 10: SSO Service Protocol ............................................................................................................................ 39
Figure 11: Authorization Service Protocol ............................................................................................................... 41
Figure 12: Administration Login Panel .................................................................................................................. 43
Figure 13: Role Registration Panel .......................................................................................................................... 44
Figure 14: Role List Panel ....................................................................................................................................... 44
Figure 15: Rule Registration Panel .......................................................................................................................... 45
Figure 16: Policy Registration Form ......................................................................................................................... 46
Figure 17: Policy List Panel ..................................................................................................................................... 46
Figure 18: Policy View Panel .................................................................................................................................... 47
Figure 19: User List Panel ....................................................................................................................................... 47
Figure 20: Application Registration Panel ................................................................................................................ 48
List of Tables
Table 1: System Security Evaluation

.................................................................52
1. Introduction

1.1. Background
Cloud computing, as a new paradigm of information technology, has been developed very quickly in recent years. The vast spread of Internet resources on the web and fast growth of service providers enabled cloud computing systems to become a large scaled IT service model for distributed network environments. Cloud computing is built on top of already existing Internet technologies and is delivered as a self-service utility. Three service models are: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Google, Microsoft Azure Platform, and Amazon Web Services are leading cloud computing vendors in the market of commercial system deployment. Regardless the utilized service model, cloud system can belong to one of the following cloud deployment models: Public, Community, Private or Hybrid[1]. The main characteristics of a cloud environment are abstraction and virtualization which make the technology to be perceived and applied completely in a different manner compared with existing traditional distributed systems. Cloud environment abstracts the implementation details of services and system from users and developers. Besides, resources in cloud computing systems become highly scalable through system virtualization which is achieved by means of resource pooling and sharing [2][3].

Integration of IT systems of organizations and businesses inside a cloud computing environment results in many technical and business advantages. However, the adoption of this technology introduces a new concern for businesses about security risks of cloud computing and indirect control over sensitive and private data. Cloud computing has all the security issues associated with distributed applications on the Internet and plus other security issues derived from virtualized and pooled resources[4]. Data storage in a cloud environment is one of the most important concerns from a security point of view. Because multiple cloud customers from the same or different organization can use the same resources or applications, certain security risks should be evaluated and solved before private and sensitive data, applications and system functionality are moved into the cloud. Multi-tenancy requires a policy enforcement mechanism, isolation, service levels, etc. Both cloud deployment model and service model have a high degree of impact on the cloud security solutions and cause different significance on multi-tenancy[5][6][7].

Cloud computing security risks depend highly on the cloud service model. IaaS model delivers computing infrastructure, physical storage, and networking as a service. Customers use those resources in order to build their desired computing platforms through platform virtualization facilities. PaaS model adds another layer on the top of IaaS and delivers platform as a service, together with application development frameworks and tools. SaaS in turn adds another layer on the top of PaaS and delivers application (software) as a service which is consumed by users via a browser or other
client program. There is trade-off between service delivery model and security solutions integrated in it. The higher service level, the more service provider is responsible for the security solutions[8].

Both cloud service providers and customers are concerned about security issues associated with the cloud environment. Although different cloud domains have different security and policy characteristics corresponding to specific functionality and usage of the system, the important aspects of secure service provisioning are generic among them. All the potential security issues associated with Identity Management, Confidentiality, Authentication, Access Control and Authorization, None-Repudiation are fatal for a cloud environment[9]. Cloud service providers try to overcome security and privacy related issues by offering security solutions to its customers. Security as a Service (SecaaS) is a new instance of cloud service model which delivers security solutions to enterprises by means of cloud-based services from the cloud. These services may be delivered in different forms, what may result in market confusion and complication of the selection process. That is why implementation of SecaaS is still limited, but usage of those cloud-based security services will more than triple in many aspects by 2013, based on the predictions made by Gartner IT research center[10].

Identity and Access Control Service should provide identity management and access control to cloud resources for registered entities. Such entities can be people, software processes or other systems. In order to give a proper level of access to a resource, the identity of an entity should be verified first, which is the authentication process that precedes the authorization process. Besides authentication and authorization processes, audit logging mechanism should be used to keep track of all successful and failed operations regarding authentication and access attempts by the application[11].

Confidentiality is achieved by different encryption mechanisms, which is the procedure of encoding data by means of cryptographic algorithms. Providing such a service will guarantee privacy of sensitive and private data and the intended entity can only decode it. Cryptographic algorithms, which are computationally hard to crack together with encryption and decryption procedures, digital signatures, hashing, certificates, key exchange and management form an encryption system which can be delivered as a service and assure confidentiality and non-repudiation in a cloud environment[11].

As such, the centralization of security services and implementation of those services through standardized security frameworks under the model of SecaaS can be viewed as an innovative and beneficial utility for a cloud environment. This approach promotes the delivery of security services to customers in a professional and standardized manner[12]. Many motives can be pointed for such kind of solution for a cloud environment: 1-aggregation of security skills and security experts, 2- effective centralized solution, 3-standardization of security practices, 4- competitive advantage in the market over the
competitors[10]. The effective management of security in cloud-based applications is one of the core factors for the successful cloud computing platform [13].

Identity as a Service (IDaaS) is one area of SecaaS and it aims to provide security services within the scope of “identity eco-system” of a cloud environment[14]. Existing cloud-based identity service mechanisms require constant improvements and enhancements as identity associated security risks have become one of the most significant issues for a cloud environment. Privacy protection for identity information is critical factor for a successful identity system [15]. The contributions of this research will be within the area of identity services for cloud environments and will be focused on designing a cloud security system which addresses current identity-related security issues.

1.2. Problem Statement
Cloud computing offers on demand services to customers with the properties of distributed systems, such as unlimited virtual resources, dynamic scalability, as well as cost advantages for business organizations. Security issues that arise within this computing environment result in various obstacles from both business and technological perspectives. There is a continuous development of security solutions with lots of challenges for a cloud environment. Security as a Service is a rather new approach to provide security solutions for a cloud environment in a professional and centralized way. Because SecaaS delivery model is very broad and not a concrete implementation and currently still in its improvement stage, few cloud providers have a system that contains centralized security infrastructure, which can provide all the needs of customers from the security perspective. Cloud-based IDaaS is not a well established practice and there is a big need of transparent and simplified cloud security infrastructure that will provide identity management services to cloud-based software services.

As a solution to this problem, this master thesis project will investigate how to manage authentication and authorization systems in cloud environments and offer an approach of cloud security system for providing authentication and authorization services to cloud-based software services through IDaaS model. At the same time, the project will focus on how to deliver those services in an interoperable and secure manner.
1.3. Purpose and Goals
The main purpose of this thesis research is to achieve a solution that provides secure and interoperable authentication and authorization systems in a cloud environment. The goals of this master thesis are the following:

- Design security system architecture for a cloud environment, which aims to deliver two identity services, such as authentication and authorization in a secure and interoperable manner, using Web Service technology. This solution will assist cloud computing platforms to provide software services to customers in a confidential, authenticated and authorized environment.
- Develop and deploy a prototype of designed authorization service that will contain the main important features and findings of this investigation.
- Provide an approach of how to build cloud security system for ensuring identity management and access control solutions for cloud-based application service providers through open and platform-independent architecture.

1.4. Research Methodology
This research follows disciplined study in order to accomplish the objectives of this investigation. Design Science research methodology was selected in order to perform this research, because master research focuses on designing and developing a security system (artifact) which addresses particular security problems for a special domain – cloud environment. This research methodology is a nominal sequence process of well defined activities according to the following referenced paper[16]. Starting with identification of the research problems and studying related solutions, existing technologies and standards, the research goals are defined. Then designing stage goes after, which leads to the preliminary solution for the entire research problem. Afterwards, prototype development process was performed. During the development stage several modifications and improvements were introduced, because of the changes in requirements and specifications. System design and prototype development is followed by testing and deployment stage. Deployment step resulted in collecting several outcomes form system functionality point of view. Finally, analysis and evaluation are performed from theoretical and practical points of view and further improvements and suggestions are presented for future work.

1.5. Thesis Outline
The report of this master thesis research comprises eight chapters, organized in the following way:
Chapter 1 presents the background of the research area and defines the motives of this investigation, the research problem to be solved, the purpose and goals required to accomplish this study, and applied research methodology. Chapter 2 presents a review and analysis of the existing solutions, areas of contributions, related standards and mechanism. Chapter 3 describes the architecture of the proposed security system, including system entities, their functionalities and security considerations. Chapter 4 describes and explains design details and specifications of authentication and authorization services of the proposed central security system, together with message protocols. Chapter 5 demonstrates the prototype implementation of the proposed security solutions. Chapter 6 provides the evaluation of the proposed security system. The evaluation is performed from two aspects: system integration and security. Chapter 7 finalizes the report by presenting conclusions from this investigation and future work which may contribute to this research.
2. Related Work

This chapter introduces and analyses some existing solutions about identity management and cloud authentication mechanisms, which are related to this research. Besides, it covers technologies and security standards utilized in this research.

2.1. Overview of Secure Identity Management Systems

2.1.1. Identity Ecosystem
The National Strategy for Trusted Identities in Cyberspace (NSTIC) in the US, which is a White House initiative, described so called “Identity Ecosystem” in order to support the enhancement of reliable, secure and interoperable identity solutions in an online environment[17]. As mentioned in this Strategy, individuals and private sectors can set up trust relationships between each other in an online environment only based on proper standards for digital identity establishment and authentication. “Identity Ecosystem” eliminates the need for individuals to manage multiple username and passwords for different online services. Individuals with a single digital identity credential can access many different online services, because these service providers trust certain third-party identity providers who manage individuals’ authentication process. Moreover, individuals can control the revelation of their private information during online authentication procedures. The Strategy highlighted four guiding principles about identity solutions in order to have an ideal “Identity Ecosystem”:

- Privacy-enhancing and voluntary identity solutions
- Secure and resilient identity solutions
- Interoperable identity solutions
- Cost-effective and easy to use

Following these principles individuals and private sectors, such as organizations and businesses, will consume interoperable, efficient, easy-to-use and secure identity solutions for online services that will maintain confidence, privacy, choice and innovation.

This type of “Identity Ecosystem” will be beneficial for both individuals and private sectors. It promises that individuals will be able to perform convenient and secure online transactions without violating their privacy. At the same time, the “Identity Ecosystem” will be an innovative platform to deploy their online businesses in order to provide attractive, practical and efficient online services to customers with trusted digital identities.
However, there are some disadvantages and vulnerabilities related to this system. Having a single credential for authentication purpose represents a single point of failure. Although users do not need to maintain number of credentials for accessing different online services, the damage is bigger when the single authentication credential is compromised. In case of losing the credential, the owner will be blocked to access all online services until the recovering of the authentication credential. Furthermore, if an attacker maliciously steals the credential, it means he or she can obtain access to all private information of the owner at different online services until the compromise issue is solved. Traceability is another issue. Even though the system is designed so that the distribution and revelation of private information is limited to organizations, it will be available to link and trace all the electronic activities of an individual with his/her digital credential. Besides, pooling huge amount of private information related to authentication credentials in identity databases will attract an attacker’s attention, because the benefit is much more in case of hacking an identity database.

2.1.2. ICAM Identity Authentication
The Identity, Credential, and Access Management (ICAM) Subcommittee, which is responsible for identity management activities of the US government, has adopted a SAML 2.0 profile which is called ICAM SAML 2.0 Web Browser SSO Profile for supporting and managing proper identity authentication during electronic transactions[18]. The ICAM SAML 2.0 Profile is based on SAML 2.0 specifications provided by the Organization for the Advancement of Structured Information Standards (OASIS). Later in this chapter the SAML standard is shortly introduced.

This Profile describes how to facilitate end-user authentication process using SAML message exchange of an identity assertion which carries authentication information in order to support online services. This Profile defines three main participants: the end-user, the Identity Provider (IdP), and the Relying Party (RP), which can be the Service Provider (SP) as well. There are two types of SAML bindings used in this Profile for exchanging SAML protocol messages. The HTTP POST binding is used to send a SAML assertion from an IdP to an RP, and the HTTP Redirect binding is used to send SAML authentication request to an IdP. The end-user establishes an identity credential with the IdP in order to request services from the RP. Once authenticated to the IdP, the end-user can access services on the RP site, as it trusts the IdP. There are two use cases defined in this Profile: IdP-initiated, where an end-user first connects to the IdP and RP-initiated, where an end-user first connects to the RP. Figure 1 shows the sequence diagram of the RP-initiated use case. This Profile allows four features based on these two use cases: Single Sign-on, Session Reset, Attribute Exchange, and Single Logout.
After obtaining SAML response message, the RP performs end-user activation. Through the end-user activation process the RP manages user’s new or existing local account with the identifier obtained from the IdP.

All security and privacy issues existing in this Profile are derived from security and privacy risks associated with the SAML standard. This Profile requires all SAML messages to be digitally signed: the RP must digitally sign SAML authentication request messages and the IdP must digitally sign all SAML response messages containing SAML identity assertions. Upon receiving a SAML message both entities should verify its digital signature. At the same time, all request-response messages should be verified against metadata. It is recommended to send SAML messages via protected channels like SSL. Messages may be also encrypted. The SAML 2.0 message exchange between the RP and the IdP requires trust relationship between them. But before establishing such relationship, these entities need to obtain certain knowledge about each other. Metadata is used to express such information. Figure 2 shows the high-level trust model for all the use cases defined in this Profile.
2.1.3. Cloud SSO Authentication

Group of researchers from the faculty of engineering of Messina University (Italy) proposed a three-phase cross-cloud federation model in order to support the establishment of cloud resource federation[19]. This model facilitates management of so called “Horizontal Federation” of cloud resources. One cloud service provider, lacking in internal resources, can cooperate with another cloud service provider in order to supplement required resources by means of external ones. The model consists of three phases: discovery of available external cloud resources, match-making selection between discovered cloud providers, and authentication for trust context establishment with selected clouds. The main focus of this model is authentication phase, which is named Cloud Single Sign-on (SSO) Authentication. Through Cloud SSO a cloud provider authenticates itself with other heterogeneous cloud providers regardless of their implemented security mechanism and accesses all needed external cloud resources. In order to establish trust relationship between home and foreign clouds, an IdP (trusted third party) is represented in this model which verifies digital identities of clouds and provides SAML authentication assertions. A new SAML profile was designed which is
called Cross-Cloud Authentication Agent SSO (CCAA-SSO) SAML Profile. Figure 3 shows the sequence diagram of the authentication process between home and foreign clouds through the IdP. For this authentication procedure two software layers are participating in each cloud site: Cloud Manager layer and Virtual Infrastructure (VI) Manager layer. The Cloud Manager layer contains Cross-Cloud Federation Manager (CCFM) software module that performs all the phases of this model by means of three software agents: discovery, match-making and authentication. The interaction between participating entities is accomplished through SAML request-response messages. First the authentication agent of the home cloud sends a SOAP request for some virtual resources to the peer agent located at the foreign cloud, which, in turn, responds with a SAML authentication request message. The home authentication agent authenticates itself to the IdP using a SSO service. Then the obtained SAML response message containing an identity assertion is passed to the VI Manager agent of the home cloud which, in turn sends the message to its peer at the foreign cloud. The VI manager of the foreign cloud, with the help of the authentication agent, verifies the SAML assertion and contacts its peer at the home cloud providing access mechanism to the requested resources.

Different cloud providers can take advantage of this profile in order to establish cross-cloud federated environment in a secure and flexible manner. However, it is a big challenge for IdPs to authenticate heterogeneous clouds willing to establish a partnership, because it requires a high level of interoperability between various security technologies. Besides, different clouds trust different IdPs. So there is another challenge to manage trust relationship between federated clouds. Finally, traditional authentication mechanisms may not be enough to secure physical and virtual cloud resources, as they are extended to different cloud providers.
2.2. Overview of Service-Oriented Architecture

Service-Oriented Architecture (SOA) is an architectural design approach for organizing and using distributed resources which may exist in different business domains. This includes methodologies and rules for designing, developing business solutions and these solutions are delivered as services [20]. As OASIS defines, SOA provides a framework for need and capability matching and for uniting capabilities to deal with needs. SOA key concepts are visibility, interaction and effect. Capabilities as solutions should be visible to needs and there should be interaction model between needs and capabilities. Basically, interaction is executed through message exchange and the effect is the result of interaction. Capabilities as services are delivered to needs in SOA. Service solutions are not domain specific or dependant. “Loosely-coupling” is the key property for SOA-based environment. The main drivers of SOA-based systems are interoperability, usability, scalability and portability [20].

Web Service standard implements Service-Oriented Architecture. World Wide Web Consortium (W3C) Working Group defines Web Service (WS) as a software system which is designed to support interoperable machine-to-machine network communication. WS has an interface which is described in a machine processable format. Other system entities consume web services through the defined interface: interaction is carried out through sending and receiving messages [21]. There are two basic architectural roles in WS-enabled environment: service provider and service requester. The interaction between the service provider and requester generally is managed through the third entity, such as the service registry, like UDDI. Figure 4 shows participants in a WS architectural model and interactions between them.

Many interrelated technologies formulate the basis of WS architecture, including XML, SOAP and WSDL [22]. The eXtensible Markup Language (XML) provides a flexible, standardized and extensible data format, which is the core factor for the ease and success of WS deployment. Platform-independent software systems may interoperate with each other through the XML serialization mechanism. The Simple Object Access Protocol (SOAP) provides a standard and extensible XML-based framework for managing and exchanging XML messages. Information is encapsulated in SOAP messages, which are
transmitted to and from WS. Different network protocols can be used to transmit SOAP messages, for example HTTP, SMTP, FTP, etc. The Web Service Definition Language (WSDL) is a XML-based language for describing Web Services in a machine-processable way. The WS description is a platform-independent document which represents all the service operations, the accessibility of service, such as data formats and the protocols, and URL to access the service.

Generally, the WS discovery process precedes the interaction process between the service requester and provider. The discovery service entity helps service requesters to find desired services. The service requester may communicate with the discovery service entity in order to locate the WS description document based on some criteria. These criteria can be some functionality, semantics or non-functional criteria like service provider name. Obtaining the WS description, the WS requester entity and provider entity agree on that document, which acts as a contract between them. There are different mechanisms to implement discovery services. Universal Description, Discover and Integration (UDDI) is a particular type of discovery service for publishing and locating WS applications. UDDI is implemented as a registry, where the service provider should actively publish its service description.

Both SOA and cloud computing are service-oriented, although SOA is narrower. SOA is focused on software as a service model, but cloud computing starts with hardware and ends up with software services. Cloud SaaS model implies Web Service development model [23]. Therefore, they share many common characteristics. Both depend on a network which should be robust and reliable. Because communication between service provider and requester is based on the underlying network, overall system performance depends on network performance. Both SOA and cloud computing are forms of outsourcing mechanism. Finally, they both provide options of common standards in order service requesters can choose for accessing and using those service capabilities through underlying network [24]. The designed security infrastructure will be completely based on Web Service technology, and interoperability is one of the main characteristics that the designed system should have. In order to manage security challenges in a cloud environment, the designed system should also be interoperable from security perspective, thus interoperable security standards will be considered in the design of both security services following the recommendations given in the referenced papers [25] [26].

2.3. Web Service Security Considerations

According to the Web Service Architecture document, provided by the W3C Working Group, security threats associated with host system, application and network infrastructure are important security considerations for WS environment[21]. Therefore, it is very important to consider them when designing SSO and authorization services.
Various XML-based security mechanisms are required to counter security risks related to authentication, role-based access control, distributed security policy enforcement, and message layer security. There are point-to-point and end-to-end security mechanisms and the choice between them is an entirely WS implementation issue. However, point-to-point security mechanisms, such as SSL, VPN, IPSec, etc., do not provide security solutions to ultimate receiver and sender. Because SOAP messages may pass through different intermediaries, end-to-end security technologies are much more appropriate for WS environment.

Three security related concepts are important in the WS architecture: resources which should be protected; protection mechanisms (policy enforcement mechanisms), and policy documents which represent constraints on resources. The following are the requirements for assuring end-to-end security in WS environments:

**Authentication** – one way or in some situations mutual authentication mechanisms should be applied in order to verify the identities of a service provider and a requester.

**Authorization** – after a successful authentication, an authorization mechanism should control access rights of resource requesters. Role-based access control and policy-based techniques can be used.

**Data Integrity and Data Confidentiality** - message information should be unaltered and inaccessible for unintended parties. Data encryption and digital signature address those issues.

**Non-Repudiation** – disagreements between a service requester and a provider about transaction occurrences should be avoided. Digital signature is the technique to protect against false denial of the transaction occurrence.

**Integrity of Transactions and Communications** – business processes and flow of operations should be executed in a proper behavior.

**End-to-End Integrity and Confidentiality of Messages** – message information integrity and confidentiality should be provided, especially when there are intermediate system entities in the message path.

**Audit Trails** – user access and behavior should be traced. Software agents can monitor and provide audit trails for systems.

**Distributed Enforcement of Security Policies** – it should be possible to define security policies and enforce them across different system platforms.
There are many Web Service Security related technologies that provide solutions to the above mentioned security problems. The W3C highlights the following technologies for securing web services:

The XML Signature standard is developed by the W3C and the IETF (RFC 2807, RFC 3275). It specifies the ability to digitally sign documents, including XML, entirely or partially. XML signatures ensure authentication, data integrity and non-repudiation [27].

The XML Encryption standard developed by the W3C details how to encrypt arbitrary data and represent the result in XML [28].

The XML Key Management Specification (XKMS) standard developed by the W3C provides XML-based way of PKI management. Together with XML Signature and XML Encryption the XKMS can be very suitable mechanism for securing web services. WS implementers have the option to outsource key registration and validation processes to a “trust” utility, thus keeping web services simple [29].

OASIS developed Web Services Security (WSS) specification which defines an end-to-end security framework to provide a SOAP extension mechanism for message integrity, confidentiality and authentication. This framework uses the XML Encryption and XML Signature standards [30][31]. Another specification describes a framework for binding SAML messages to SOAP protocols [32].

The SAML and XACML standards are briefly introduced in the next sections of this chapter.

### 2.4. Overview of the SAML Standard

Security Assertion Markup Language (SAML) is an XML-based standard created by the OASIS Security Services Technical Committee. The purpose of the SAML standard is to describe and exchange security information via SAML assertions between online business domains that trust each other. This standard has strict syntax and rules for managing SAML assertions. The SAML is the core standard used for designing cloud authentication service in this project and the design is based on cloud authentication frameworks described in the following referenced papers[33][34]. The SSO mechanism benefits from the usage of the SAML standard, which provides a solution to transfer security information independent of any specific platform, domain and protocol. The following subsections briefly introduce SAML standard’s main features and rules; complete specifications and details can be found in the OASIS document of SAML V2.0 technical overview[35].
2.4.1. SAML Concepts and Components

As already mentioned, SAML standard facilitates exchange of security information, like identity, authentication and authorization data, between different business domains or entities. SAML specifications contain different standard components that define all the necessary steps and concepts for SAML management. At least, there are two parties that take place in security exchange scenarios: SAML asserting party and SAML relying party. Assertions carry statements that contain security information about a subject claimed by the asserting party. The assertion subject is an entity, such as human or system entity, to which the assertion is addressed. Assertions contain assertion statements, such as authentication statement; attribute statement; and authorization decision statement. Assertions are transferred via SAML request-response messages between different trusted business domains. SAML protocols define those messages. Low–level transporting protocols, like HTTP or SOAP, are used to map SAML messages into low-level messages and transfer them between independent domains. SAML bindings define the mapping mechanisms of SAML messages into low-level communication messages. Another important SAML component is SAML profiles, which define various business scenarios and restrictions on SAML assertions, protocols and bindings, as a solution for those scenarios. Figure 5 shows relationships between those components. SAML standard is quite flexible and is not limited to above mentioned components.

![SAML Components](image)

Figure 5: SAML Components [35]

2.4.2. SAML Privacy and Security Features

The OASIS specification provides deep and extensive description and analysis about privacy and security properties of the SAML[36]. As SAML message carries security information in assertion statements, confidentiality of those statements is a privacy issue.
Entities, which provide, consume or exchange SAML assertions, need to consider risks associated with privacy problems. SAML standard applies “anonymity” as an approach to solve privacy issues for SAML subjects. The environment that consists of SAML-based systems is limited to “partial anonymity” at the best, since authorities (attesting parties) have relationship with those subjects. Pseudonyms are used to hide subject identities. However, the reuse of the same pseudonym can result in an identification of the subject. One-time-use pseudonyms promise anonymity of the subject identity. In SAML-enabled environments even an anonymous subject can be identified, based on repeated unusual behavior. So user behavior should not be traceable.

The communication environment of multi-domain SAML system entities is subject to many security risks. That is the reason why correct implementations of security protocols are essential for security of the whole system. It is also very important to establish trust model between related system entities. Potential threats are the following:

- Collusion between two or more system entities to execute an attack
- Denial-of-Service attacks
- Man-in-the-Middle attacks
- Replay attacks
- Session hijacking

Different local mechanisms are used to make a decision of assertion generation considering the above mentioned threats and those approaches are specific to system implementations. Because SAML standard provides means to exchange security context via SAML messages between multiple domains, authentication, confidentiality, data integrity and non-repudiation become the key security properties that should be assured. In a SAML-enabled environment two security PKI-based mechanisms are applied: the first one is to use secure communication channel via secure network protocols like SSL, TLS or IP Security Protocol; the second one is to use message level technical solutions like XML Signature.

### 2.5. Overview of the XACML Standard

eXtensible Access Control Markup Language (XACML) is a XML-based standard created by the OASIS XACML Technical Committee. The motivation for the creation of this standard is to provide a common policy language in order to define security policies for access control decisions in multi-domain scenarios. The XACML is the core standard that is used for designing authorization service in this project, and more particularly, the service is designed following the approaches of policy management mechanism, described in the following referenced papers [37][38][39]. In a multi-vendor environment interoperability between authorization implementations can be reached through the use of
XACML standard. Security policy management has many phases starting with writing policies, ending with enforcing policies. The complete specifications and details can be found in the OASIS document of XACML V3.0 core specification [40]. Figure 6 shows XACML context: in a multi-domain environment XACML resources are separated from the application environment by XACML context. This means that domain specific inputs are converted into XACML context representations, which are operated and then converted back to domain specific outputs.

![Figure 6: XACML Context [40]](image)

XACML standard is designed primarily for Attribute-Based Access Control (ABAC) systems; however, it supports also specialized implementations of ABAC like Role-Based Access Control[41].

### 2.5.1. XACML Components

There are four main conceptual components in an XACML-enabled environment, which are correlated in order to manage the complete XACML functional model for the system: PDP, PEP, PAP, and PIP.

Policy Decision Point (PDP) is a system entity responsible for storing policies, analyzing policy information upon request, making decision and sending it to the PEP. Policy Enforcement Point (PEP) is a system entity responsible for executing access control based on authorization decisions made by the PDP. Policy Administration Point (PAP) is a system entity that manages the process of creating policies. Policy Information Point (PIP) is a system entity that is responsible for providing additional attribute values, which may represent characteristics of environment, actions, resources, etc. Two or more system entities may be combined in one computing node and besides, they may communicate with each other through a repository system. The interaction between components is accomplished via XACML request-response protocols. XACML request-response protocols may be mapped into application specific protocols, such as SAML, meaning that SAML request-response messages can be used to transmit XACML request-response messages. The XACML policy language model consists of three main elements: Rule, Policy and Policy Set. The rule is the main component for a policy,
because it represents the basis for decisions made by the PDP. In turn, the rule consists of elements, such as target, rule effect, condition, obligation, and advice.

2.5.2. XACML Security and Privacy Considerations

XACML-based systems are subject to security and privacy related treats which should be considered during the implementation phase. Because the XACML model is based on interactions and dependencies between the XACML components, it is very important to establish trust relationship between them specific to the concrete system implementation. XACML V3.0 core specification documentation [40] highlights those compromise situations significant for XACML-enabled environments:

- Unauthorized disclosure
- Message replay
- Message insertion
- Message deletion
- Message modification

As the XACML documentation states, two approaches can be applied to solve the above mentioned security issues, which may be combined together according to the system sensitivity. The first one is to use communication level security mechanisms like SSL; the second approach is to use message level techniques, such as XML encryption.

Besides the channel-level protection, there should be also protection at the storage level, meaning that all the policy files stored in repositories should be protected and checked before to use. What mechanism to apply is subject to a particular system implementation; however, policy confidentiality and integrity are very essential for secure access control systems.

This chapter describes the architecture of a cloud security system, which is designed for delivering authentication and authorization services to cloud-based application service providers. Those services are designed taking into account the solutions and security standards outlined in the second chapter.

3.1. Overview of Cloud Architecture Model

Figure 7 shows logical representation of a cloud environment: different applications running on a cloud platform deliver different services to end-users through the Internet.

End-users can be people or other business entities; however, regardless of the user type, communication channel should be secure. As the picture shows, users interact with an access point entity through the Internet. Here, the access point is the logical representation of a cloud access point acting as an entry point to cloud-based services (practically, there is not only one access point, but application servers run behind different types of access points and proxy servers). As the communication is managed through message exchanges, there may be a need for message encryption, decryption, digital signature, etc., according to enterprise requirements and needs. This means that both communicating parties will need public-private key pairs to protect resources; and as different service providers (enterprise entities) may exist and run their applications within the same cloud environment, appropriate PKI system should be adopted for that environment.
3.2. Security System Design Approach
As mentioned in the previous chapter, SOA implementations, such as Web Services, are focused on the definition of services and delivery of those services in an interoperable manner between multi-domain environments. Practically, those services are software blocks that are capable to provide some functionality and business logic to service consumers through a service delivery model. In order to provide security solutions as services, there is a need to define an architecture consisting of entities that handle the security system functionality. Those system entities act as security service providers, which constitute secure environment for cloud-based systems. This research focuses on two types of security services: authentication and authorization. Both services are implemented using Web Service technology and interoperability is one of the main features of the designed security system for delivering those services.

3.3. Overview of the System Entities
In order to provide authentication and authorization services according to the cloud SecaaS model, there is a need to establish a separate security infrastructure within a cloud environment. This means that all identity services will be delegated from individual application service providers to the shared security system. In this way the application service providers can focus only on their business logic, rather than implementing built-in authentication and authorization services. Both security services depend on the Identity Management System (IDMS) and PKI system. IDMS provides registration services. All the entities need to register themselves before they can use those identity services. PKI system has a very important role for cloud-based systems: it provides X.509 certificate services and establishes trust relationship in the cloud environment. This ensures machine-to-machine secure communication between system entities, as well as the privacy of information stored and exchanged between them. Besides, the PKI becomes the basis for authentication and authorization services. In our security architecture model we assume that reliable IDMS and PKI system entities are already available.

Figure 8 shows the model of the central security system architecture taken as a whole. There are three servers, such as IDMS, central authentication server, and Local Certificate Authority (LCA) server that are necessary for the authentication and authorization services in order to provide appropriate identity solutions. Central authentication server is responsible to manage authentication procedures, and in addition to that, it also acts as a proxy server for all the service providers in the central security system. SAML entity is in charge of providing authentication service and PDP entity is in charge of providing authorization service for a cloud environment. The SAML and PDP system entities can be deployed at the same server. Central security system is completely secure environment and it is only controlled by cloud security administrators.
Figure 9 shows security infrastructure for a cloud environment. According to this infrastructure, there are two groups of security servers for a cloud environment: central and portal. Central security servers are the ones that are residing in the central security system and they ensure the main functionality for both authentication and authorization services. Portal security servers are residing in front of a web portal, which acts as a gateway to cloud-based application services. The role of portal security servers is to consume security services delivered from the central security system and based on that, safeguard cloud-based application services. There are two portal security servers: proxy server and PEP server. These two servers can also be grouped as a single entity, as it is an implementation issue. PEP server is in charge of delivering protected application resources and services only to authenticated and authorized users according to the decisions made by identity service providers. For our system we have adopted SAML-conformant PEP, meaning that PEP entity communicates with other system entities in the central security system using SAML-based request-response messages. By isolating security services from web portal domains, we are grouping security skills under a centralized control. Although each application service provider secures its web portal using proxy server and PEP server, it does not manage end-user authentication procedures and it does not maintain an authorization decision point. SAML and PDP system entities, as shown in the Figure 9, deliver identity services to application service providers in a secure and interoperable manner.
3.3.1. Authentication System

A single enterprise may provide many application services to end-users. E-mail servers and web servers are examples of application services providers. As company’s boundaries broaden, the number of application services grows. Mostly all service providers should authenticate clients before service transactions are executed, because they are dealing with personal information. This means that the client should have security context for each application server and log in before it can consume any service. The same situation happens when the client accesses resources in different security domains. As mentioned in the second chapter, having many security credentials for authentication purposes is not an effective solution from security, system coordination, and management perspectives. While organizations migrate to cloud environments, the same problem still exists.

To this problem, as a solution a Single Sign-on (SSO) protocol is proposed, which is part of the shared security system of a cloud environment. This solution relies on the SAML web browser SSO profile, which complete description can be found in the following referenced document[42]. The system consists of a SAML server which provides SSO services for application service providers: SAML server issues SAML ticket which contains an assertion about the client’s identity verification, thus confirming that it has
been properly authenticated or not. Once the user is authenticated, he or she can request access to different authorized resources at different application provider sites without the need to re-authenticate for each domain. As shown in Figure 9, SAML server resides in the shared security system. Besides SAML assertions issuing server, there are three other security entities in the central security system, coordinated with each other, in order to accomplish the desired solution. When the user wants to access some resource at some application service provider site for the first time, he or she is redirected to the central authentication server by the PEP running in front of the application service. The central authentication server makes identity verification according to the Strong Authentication Protocol specified by the Federal Information Processing Standard (FIPS) 196[43]. It can be one way or mutual authentication process. Authentication server verifies whether the user is registered in the IDMS database. In case of unregistered user, the authentication process is terminated and the server notifies that the user is not registered in the IDMS. If the user has a valid registration entry confirmed by the IDMS server, his or her X.509 certificate is verified in cooperation with the Local Certificate Authority service. The result of the authentication process is passed to the SAML server which, in turn, issues a SAML ticket confirming whether the user is authenticated or not. SAML ticket has a validity period which is calculated according to the system policy. SAML ticket is passed to the user (client application) through the authentication server. Then the ticket is embedded in the request directed to the application service provider. The request message is intercepted by the PEP, which verifies the embedded SAML ticket. Once the ticket confirms that the user has been successfully authenticated, a valid local session is created for the user. Until the validity period expires the user can request services from other application service providers with the same ticket without re-authenticating himself. This mechanism works because there is a trust relationship between the SSO service provider and application service providers existing in different security domains. All application services should be registered in the IDMS in order for the SAML server to deliver SSO services to them. At the same time, SSO service provider also needs to register itself in the IDMS. The IDMS server provides registration services to identity service providers, thus making them available to be looked up and consumed by the application service providers. SSO service provider publishes its metadata, which contains the WSDL or the WSDL URL, in the IDMS. PKI system establishes a trust relationship between application service providers and identity service providers. 

As the SSO system is designed using WS technology, other cloud providers which lack such identity services can benefit from it. Foreign clouds can register themselves in the IDMS as external cloud platforms and consume SSO service in favor of their cloud environment. In this case, the IDMS service should provide identity federations services. Identity-related information is outside of the SAML message exchanges. When the SAML request message is delivered to the SSO service provider, the latter first checks whether the service requester is a trusted entity with the help of the IDMS service
provider. It is up to the IDMS service provider to check its registration validity, either locally or in a federated environment. The same approach can be applied when the subject’s registration validity, to which the SAML assertion is addressed, needs to be verified. Top root CA establishes a trust relationship between two clouds. Integration details with other cloud providers are out of the scope of this research.

3.3.2. Authentication System Security Protection

SSO service provider interacts with service consumers through request-response message protocols. All system entities securely store their private keys locally. SAML server issues tickets according to the decision made by the central authentication server. That is why they communicate only over trusted internal network. At the same time central authentication server communicates with the IDMS and CA servers over a trusted network. Therefore, the central security system is an isolated secure environment, where all the system entities trust each other.

However, the issued SAML ticket is transmitted over a public network (Internet) to the end-user and from the end-user to the application service provider. Because of the potential security risks associated with SAML-based environment, the SAML ticket, during transfer, should be protected against threats, such as replay attacks, message modification, message information disclosure, impersonation and repudiation. Application service provider relies on the decision made by the SSO service provider. That is why there is a pre-established trust relationship between them based on a PKI model. The LCA server issues X.509 certificates to all type of service providers, such as SSO service provider, application service provider, IDMS service provider, etc. There is a root X.509 certificate belonging to the LCA, which is used to verify all the certificates issued for different service providers in a cloud security domain. All response messages delivered by the SAML server are digitally signed using XML Signature. XML Signature ensures end-to-end secure communication between SSO and application service providers. Application service provider verifies the signature using the certificate of SSO service provider. In this way it is ensured that the ticket has not been modified during transmission and it has been definitely issued by the SSO service provider. The ticket is a security token, and in case of a stolen ticket, an adversary can impersonate the user to the application service provider. The stolen ticket is also subject to replay attacks. In order to prevent such security issues, the traffic between the end-user and application service provider takes place only over a secure channel, such as SSL/TLS. In the same way the traffic is secured between the end-user and authentication service provider. SSL ensures point-to-point secure communication between communicating participants. Besides, SAML server refuses to issue an assertion ticket for the same assertion ID contained in the redirected request more than one time.
In some situations privacy of user identity information should also be considered. SAML server may use data privacy mechanisms provided by the SAML standard. User anonymity is ensured by means of pre-established pseudonyms between the SSO service provider and service consumers. It issues also one-time identifiers which prevent users from being tracked by application service providers.

3.3.3. Authorization System

As already mentioned earlier, different application services may be hosted in a cloud environment and may use the same physical resources. However, each application service is logically separated from others. Different types of system entities consume those services; therefore, application service provider should manage a proper mechanism for access control decisions. This means that various users, after being successfully authenticated, should request and access those resources and services for which they are authorized in a particular enterprise security domain. As the number of the services and service consumers grow, management of access control mechanism becomes more complex and expensive: each service provider needs to implement independent access control mechanism by means of self-governing security policies and policy enforcement points. Decoupling policies from application services and managing them independently from application services results in a solution which is more effective for an authorization system. Applications focus only on system functionality and business value. Having a single security policy management point makes the entire authorization system more flexible and secure, meaning that it can be administered, configured and protected separately from application services. In this way, it is easy to configure and apply common policies for every application service in a single security domain. Besides, changing a policy becomes very simple because of a single location for policy management. Protection and auditing of the authorization system is managed separately thus making it much harder to compromise.

Role-based authorization system is proposed for a cloud environment which is a component of the central security system. XACML is the main standard adopted for this authorization system. The system provides authorization services for cloud-based application services. As shown in Figure 9, Policy Decision Point (PDP) server resides in the central security system. It implements role-based access control mechanism and provides authorization services to application service providers within a security domain. Policy Administration Point (PAP) component is in charge of providing policy administration services to security administrators. It is the main repository for policies and authorization service provider makes authorization decisions based on security policies created and stored in that repository by security administrators. In the designed security system PAP component is deployed in the PDP server. End-users, that may access resources at an application service site, must be assigned different access roles by
security administrator. PAP provides role defining and assigning services to authorized security administrators. In order to assign a role to an end-user, the latter should have a valid registration entry in IDMS. PAP and IDMS are coordinated together and they share a repository for storing and retrieving end-user attributes, such as roles. At the same time, security administrator defines role-based policy: it represents authorization result based on a combination of resource, action and role. Thus, the complete decision service is centralized in a single security system. XACML policy language is used for creating policy files. As already mentioned, according to the proposed cloud security infrastructure shown in Figure 9 application services are protected by the PEP server. When a user sends a request to access some resource or service, PEP server intercepts user’s request and creates an XACML authorization request. Then the request is sent to the PDP service provider. PDP service provider makes an XACML authorization evaluation against already created policies and returns an authorization decision result back to the PEP. PEP server is responsible to enforce the authorization decision through granting or denying the access for a particular resource or service. As in the case of SSO service, authorization service metadata, which contains the WSDL or the WSDL URL, should be published in IDMS. PKI system establishes a trust relationship between PDP, PAP and PEP components.

### 3.3.4. Authorization System Security Protection

Application resources and services are located mostly in a separate enterprise domain within a cloud environment. PEP is responsible for controlling access to these resources and services based on decisions provided by the PDP. PEP communicates with PDP through a public network using request-response message protocols, as shown in Figure 9. Although PDP service is running inside trusted central security system, it is still subject to potential security threats associated with XACML-based environment, such as replay attacks, message modification, message information disclosure, impersonation and repudiation. Therefore, communication channel should be secured. Policy administration procedures should also be protected.

In the proposed authorization system PDP and PEP services mutually authenticate each other before any service transaction occurs. Authentication process is performed at a message level using digital signature technique. All XACML messages are digitally signed using XML signature standard in order to ensure that they originated from the intended entities: PDP and PEP. PDP and PEP components establish a trust relationship between each other based on the underlying PKI model. If message confidentiality is necessary for particular system implementation, XACML request-response messages can be encrypted using XML encryption standard. Randomly generated session IDs in messages help to avoid replay attacks, because PDP server refuses to make an
authorization decision evaluation for the same assertion ID contained in the XACML request message more than one time.

The PDP service totally depends on the PAP service. The repository where all policy files are stored should be securely safeguarded. Only authorized system entities and users, such as security administrators, have access to the policy repository and PAP services after successful authentication process. Authentication process is implemented using FIPS 196 Strong Authentication Protocol. Upon creation of a policy, PAP service digitally signs it using XML signature standard. This ensures the integrity of the policy that the PDP service should evaluate for making an authorization decision: PDP verifies digital signature to be sure that it has not been modified after originally created by the PAP service. In some scenarios policy files may contain sensitive data, therefore they need to be encrypted using XML encryption standard.

### 3.4. Summary

This chapter has introduced the architecture of central security system which is designed for a cloud-based environment. The system is designed using service-oriented architectural approach: the system delivers security services, such as authentication and authorization services. This chapter has highlighted the significance of those security services in a cloud-based environment. It has also mentioned all the potential security threats that may compromise the system, as well as the necessary mechanisms and solutions to prevent them. Central security system provides flexible and effective security solutions in an interoperable manner. Those solutions are easy to consume by service requesters as they are designed based on the WS technology.
4. Design and Specifications of Authentication and Authorization Services

This chapter describes the design details of authentication and authorization services and their corresponding communication messages and protocols.

4.1. Design of Service Interfaces

The proposed shared security system is designed using WS technology. All system entities in the shared security system act as service providers and deliver security related solutions to cloud-based entities. They have well defined interfaces which enable service requesters to consume those services without any complexity. Each service has an input parameter (some services may not require an input parameter) and corresponding output parameter. These parameters conform to request and response messages for each service. The request-response messages are wrapped into the XML format, thus making platform-independent system entities to interact with each other in an interoperable manner. Each service provider has a description of provided services, which is remotely available to service requesters. In this system, the security service providers register their services and publish their WSDL URLs at the IDMS. The application service provider looks up for the desired service at the IDMS service provider and obtains the URL of the WSDL file for that particular identity service. Then the application service provider obtains the WSDL document and based on that description the service can be easily consumed.

4.2. SSO Service

As described in Chapter 3, SAML server provides a SSO service to application service providers. The end-user authentication process is completely controlled and managed by the central security system of a cloud environment. For this system all SAML messages are transmitted using the HTTP-Redirect or HTTP-POST binding. In order to get a SAML ticket, the PEP server needs to connect to SSO service provider endpoint for incoming requests and call the Request_SAML_Ticket service. Through this call it sends a SAMLAuthenticationRequest message to the SAML server. The message is directed to the SAML server through the central authentication server which acts as a proxy server. The latter intercepts the message. As the request message is for authentication purposes, it starts to authenticate the end-user. The authentication result and SAMLAuthenticationRequest message are passed to the SAML server. In turn, SAML server issues a SAMLAuthenticationResponse message based on the authentication result and request messages. The SAMLAuthenticationRequest message must contain assertion ID for a particular message, ID and service URL of the service requestor; in this case the ID and service URL of the application service provider. The ID must match the
registered ID in the IDSM database and service URL must match the one described in the service metadata. The message also contains assurance level of identity parameters for the authentication process: identity verification will be at that level. There may be other elements included in the request message. At the end, the message should be digitally signed by the service requester. The authentication result contains the subject ID according to the requested format and the status code and value of the identity verification process. The SAMLAuthenticationResponse message must contain assertion ID of the request message, ID and service URL of the SSO service provider, authentication result status, and assurance level of performed identity verification. There may be additional elements included in the response message. Before sending back the response message, it should be digitally signed by the SSO service provider.

4.2.1. SSO Service Protocol

Any user or client application, before accessing any resource provided by the application service, is first required to be authenticated. The SSO can be IdP-initiated or RP-initiated in our system and the authentication process contains multiple interactions between different system entities. Figure 10 shows communication protocol between participating entities in the SSO service. This is a RP-initiated SSO. The end-user first connects to the application service provider through request resource message in order to request access to a protected resource or service. The request message is intercepted by the PEP server. If the end-user does not have a valid local session for that particular application service, PEP returns an authentication request message, such as SAMLAuthenticationRequest and directs the end-user to the SSO service provider. The user connects to the strong authentication server via HTTP Redirect message protocol. Then the authentication process is executed according to the Strong Authentication Protocol provided by FIPS 196 specification. In the protocol diagram the authentication server authenticates only the end-user and it is based on the user’s X.509 certificate. In some cases a user may also authenticate the authentication server. Then the user’s identity registration is verified with the IDMS service provider. Besides, the authentication server communicates to the LCA server in order to check the validity of user certificate against certificate revocation list published by the LCA service provider. Getting the certificate verification result, the authentication server requests the SAML server to issue a SAML ticket. The SAMLAuthenticationResponse ticket is returned to the user through the authentication server according to the HTTP Post message protocol. Then the user is redirected to the application service provider. The response message is intercepted by the PEP, which verifies first the ticket validity and then may grant the user access to authorized resources or services based on that SAML ticket. More specifically, if the user has been successfully authenticated, then PEP creates a local valid session.
In case of any SAML ticket issuing failure, the service returns a response message containing the failure information together with the failure status code.

4.3. Authorization Service

After successful authentication, the user may request protected resources or services. As described in Chapter 3, central security system is responsible for authorization decisions as well. The PDP server delivers a single service which provides authorization decision based on XACML policies. The service requester (PEP) needs to connect to the PDP service endpoint, obtain a reference to the service object, and call the \textit{Request\_XACML\_Authorization\_Decision} service. Through this call PEP server communicates with PDP service provider using authorization decision request and authorization decision response messages, which are in XACML format. As mentioned in Chapter 3, we have adopted SAML-conformant PEP for our designed security system. Therefore, authorization decision request and authorization decision response messages are embedded in \textit{SAMLAuthorizationRequest} and \textit{SAMLAuthorizationResponse} messages. Because PDP service provider makes authorization decisions based on policy files in XACML format, there is a need to map each \textit{SAMLAuthorizationRequest} message into XACML request context and XACML response context into \textit{SAMLAuthorizationResponse} message. In order to map and transfer XACML-formed
request-response messages in SAML-based messages, SAML profile of XACML should be used. The complete specification of this profile can be found in the following referenced document [44]. It is the PEP’s responsibility to protect resources from incoming requests and initiate an authorization evaluation process. The SAMLAuthorizationRequest message must contain assertion ID for a particular message, ID and service URL of the application service provider. The ID must match the registered ID in the IDSM database and service URL must match the one described in the service metadata. The message should be digitally signed by the PEP. The SAMLAuthorizationResponse message must contain assertion ID of the request message, ID and service URL of the PDP service provider. Before sending response message back to the PEP, it should be digitally signed by the PDP service provider.

4.3.1. Authorization Service Protocol

PEP must control access to different application services, when user or any other system entity requests access to protected resources or services from the application service provider. Figure 11 shows the communication protocol for the authorization service. The user requests access to a resource at the application service site. PEP server intercepts the request message and constructs a SAMLAuthorizationRequest message including XACML authorization decision query statement which contains the requested resource URL, action and the role of the user. As it is role-based authorization system, each application service administrator must assign roles to users. As already mentioned in Chapter 3, PAP provides user role assigning service and IDMS provides user attribute retrieving service. In order to obtain user role PEP must query IDMS. SAMLAuthorizationRequest message is sent to the PDP service provider. Upon receiving the message, the PDP service provider makes the authorization decision for that particular request and returns the XACML authorization decision result back to the PEP in a SAMLAuthorizationResponse message. The message contains the decision status code and one of the four XACML decision values: Permit, Deny, Indeterminate or NotApplicable. If authorization decision evaluation is successful, meaning that PDP has located the only registered policy for a particular XACML request, then the result contains target rule effect, such as deny or permit, defined by the security administrator. If there is no applicable policy for a particular XACML request (role, resource URL and action), then the result contains a NotApplicable decision value. In case of any XACML authorization decision issuing failure, the service returns the result as indeterminate together with the failure status code. Indeterminate decision value is also returned when PDP has located more than one policy for a particular XACML request. If the same policy contains two identical rules (even if rule effects are different), then the authorization decision result is evaluated against the first encountered target rule.
In addition, the response message may contain an obligation or advice element. The PEP enforces the authorization decision: it either permits or denies the access. In case of permit, the application service returns the requested resource.

![Diagram of Authorization Service Protocol]

**Figure 11: Authorization Service Protocol**

### 4.4. Summary

This chapter has presented the design of the security services of the proposed central security system and corresponding request-response message protocols. The SAML entity provides one service: SAML ticket-based Single Sign-on service. The PDP entity provides also one service: role-based authorization service. All these security services are designed using WS technology approach which enables loosely coupled distributed system entities to interoperate flexibly with each other.
5. Prototype Implementation

This chapter describes a prototype implementation of the authorization service based on the proposed design of the authorization solution for a cloud environment. The purpose of the prototype is to demonstrate and test the functionality of the authorization service provider. The implementation of the authorization system consists of two parts: Policy Administration Point (PAP) Service and Policy Decision Point (PDP) Service. The PAP service manages a role-based access control mechanism for security administrators and based on that service, the PDP service manages an authorization service for cloud-based system entities. As described in the previous chapters, the authorization service model is designed completely through the SOA approach, specifically using SOAP-based Web Service technology. The implementation is built on Java platform using already existing and tested libraries and software frameworks. Applying Web Service technology to our authorization service implementation, it makes the system interoperable at a high level together with the ease of deployability, usability and system integration. Besides, Web Service technology provides all the necessary security mechanisms in order to manage a secure environment between service requesters and providers.

5.1. Policy Administration

The implemented PAP service is used by system security administrators for managing and administrating role-based access control policies for their application service environment. It provides user friendly web interface for policy administration which is implemented using JavaServer Faces (JSF) Model-View-Controller web framework and deployed in Apache Tomcat and MySQL servers.

The Apache Tomcat service provides runtime environment, more specifically, a container for the PAP service provider to deliver policy administration services to security administrators through http/https protocols.

The MySQL database is used to store roles, rules, policies and other objects registered by administrators. Registered policies are also stored in the form of XML policy files. The Hibernate Java persistent framework is used for managing and querying the database server.

Security Administrators can register, view, update or delete objects such as roles, rules and policies. These actions conform to the four basic functions of persistent storage, such as create, read, update, and delete (CRUD). Besides, they can assign roles to already registered users and register application information. This PAP prototype implementation allows only registering one policy per each resource URL. The following subsections describe all functionalities provided by the PAP service.
5.1.1. Access to the PAP service

Before accessing any service provided by the PAP system, the administrator must authenticate himself to the PAP service provider. Figure 12 shows the login page: administrators are authenticated using username and password credentials which are defined and distributed to them beforehand in a secure manner.

![Administration Login Panel](image)

Figure 12: Administration Login Panel

Browser is used as a client application to connect to the PAP service via the Internet, so the communication channel is secured using SSL over HTTP, through which the service authenticates itself to the browser. This means that all the messages that are transmitted between the browser and PAP service provider are secured: SSL ensures the integrity and confidentiality of the messages transmitted between authenticated end-points, such as the browser and PAP service.

After successful authentication, the security administrator can use all the services that are provided by the PAP system.

5.1.2. Creating and Managing Policies

As the authorization service is based on the role-based access control mechanism, administrator can register a role using a web interface. Besides, already registered roles can be updated or deleted. Figure 13 shows role registration panel: the administrator, first, selects the “Roles” object on the left side of the web page and then the “Register” function. The role has three properties: name, description and domain. The role domain uniquely identifies the role within the scope of its usage.
If the administrator selects “List” function, all registered roles are listed. Furthermore, there are two corresponding functions for updating and deleting purposes. As Figure 14 shows, when “List” function is selected, it displays all the registered roles which can be then updated or deleted. There are nine roles which are defined and hardcoded in the system beforehand for our cloud environment and during system initialization these roles become registered. Policy files contain role information for our role-based authorization system that is why it is necessary to register roles before defining policies.

Because each policy file contains at least one rule, it is necessary first to define rules and then, include them into a policy file. Figure 15 shows rule registration panel. When “Rules” object is selected on the left side of the web page, two functions are enabled: Register and List. In order to register a rule, the administrator should define it. As we have adopted the XACML standard for our policy-based authorization system, the “Rule”
object contains the least required elements in order not to break the XACML policy structure. Rule has a “Name” attribute and its value should be unique; otherwise, the rule will not be registered. Each rule has also an “Effect” attribute and its value can be one of the two options: Permit and Deny. The administrator may provide a description for particular role. The “Subject Type” is a drop-down list for selecting the type of the subject to whom the rule is addressed. Finally, “Action” element should be selected from the drop-down list as it defines the operation for which the rule will be evaluated.

If “List” function is selected, the list of registered rules are displayed which may then be updated or deleted.

After defining the roles and rules, the administrator may register policies. Upon policy registration, an XACML policy file is created, digitally signed by the PAP entity using its private key and stored in a secure repository. For this implementation, policy files are not encrypted, although in some cases it may be required

In order to register policies for the authorization system, “Policies” object should be selected. Again, “Register” function button displays the registration form. The administrator gives the name of the policy, which should be unique; otherwise, it will not be stored in the repository. The value of “Rule Combining Algorithm” attribute should be one of the three given options. Each policy may have a “Condition” element that narrows the scope of policy evaluation. As we have a role-based authorization system, the value of the “Condition” element is one of the already registered roles which can be selected from the drop-down list. The “Resource URL” element defines the target of the policy file for which it will be evaluated. Finally, the policy will contain at least one of the already registered rules. Upon form submission all the input values will be validated and
an XACML policy file will be created. The policy file will be digitally signed using XML Signature standard and then stored in the repository. Thus, it is protected from malicious modifications. Figure 16 shows the policy registration form.

Later, the administrator may update or delete already registered policy file. It is also possible to view XAML policy file in the XML format by calling "View" function. Figure 17 shows all the registered policies displayed after calling "List" function.

As Figure 18 shows, the registered policy file is displayed in XML format by calling "View" sub-function. This gives the possibility to inspect the content of the policy file excluding the digital signature. An example of digitally signed policy file can be found in appendix B. The digital signature for the policy file is calculated using RSA private key. SHA-2 algorithm is applied for message digest and signature.
5.1.3. Assigning Roles
In order to assign a role to a user, the administrator should select “Users” object on the left and call “List” function, which displays all registered users in IDMS. As shown in Figure 19, security administrator should select a user and then call one of the sub-functions: “Assign Role”, “Update Role”, “Delete Role”, and “Home Page”.

Figure 19: User List Panel
The PAP service can assign already defined roles to registered users, because it shares a repository with IDMS for storing, updating or deleting user attributes. This prototype system does not allow assigning more than one role to user.

5.1.4. Registering Applications
This prototype service allows registering application service data. Generally, IDMS is responsible for providing application service registration services. However, the available implemented IDMS does not provide such service at the moment of this prototype implementation. When “Applications” object is selected on the left, two functions are enabled: “Register” and “List”. Figure 20 shows application registration panel. Calling “List” function, all registered applications are displayed, which can be read, updated or deleted.

Figure 20: Application Registration Panel

5.2. PDP Authorization Service
The PDP service is implemented using SOAP-based Web Service technology on Java platform. XACML open source Java libraries provided by JBoss community are used in this PDP service implementation.

The PDP service is running inside a Tomcat container as a runtime environment. The Web Service approach makes authorization service requesters, such as PEPs, very easy to consume it. PDP service description, as a WSDL file, is published in the same container. WSDL file of this service can be found in appendix A. Using this WSDL file, the PEP entity can easily connect to the PDP service provider and request the authorization service. This PDP service provider makes accessible only one service: Request_XACML_Authorization_Decision. The service requires an XACML request
message that contains three attributes: resource, action, and role and produces an XACML response message, that contains the authorization decision together with the corresponding status code. For this prototype implementation XACML request-response messages are not mapped into SAML-based message protocols. Both XACML request and response messages are embedded in the body of SOAP messages. SOAP messages are serialized in the XML format during the transmission.

In order to make an authorization decision evaluation, PDP service loads all the registered policy files from the repository. Before evaluating any decision, each and every digital signature of the policy files is verified by the PDP service provider using the public key of the PAP entity. If the digital signature verification fails, the policy file will not further be considered for the evaluation. Thus, the authorization decision evaluation is based on policy files that have not been modified after originally created or updated by security administrators. The communication channel between the PDP service and PEP entities is also secured though enabling the web service security mechanism called Mutual Certificates Security. This guaranties mutual certificate-based authentication between the PDP service and PEP entity, as well as message integrity and confidentiality exchanged between them. The service description includes also all the security related information that is required from service requesters in order to establish a secure communication.

5.3. Summary
This chapter has introduced the implemented prototype for the proposed authorization service. Two services were implemented: PAP and PDP. The authorization system was built using XACML role-based access control policies. SOAP-based Web Service technology was applied for authorization service prototype implementation combined with built-in security mechanisms. The functionality and components of the implemented authorization system were described and explained.
6. System Evaluation
This chapter presents the overall evaluation of the proposed security system from two perspectives: integration and security. Integration demonstrates how the proposed security services can be integrated within a cloud environment. Security demonstrates how securely the services are delivered to service requesters.

6.1. Web Service Security and Integration Advantages
Both SSO and authorization services are designed using Web Service technology. As mentioned in the first chapter, cloud computing platform is completely service-oriented and is accessed through high level Web API. That is why the integration of these security services within a cloud environment does not cause technology incompatibility issues. Moreover, it can effectively be deployed and exploited through utilizing all the benefits of service-oriented architecture.

Here are the main Web Service advantages that the proposed cloud security system obtains from the service-oriented architectural design:

1. **Loosely coupling** – the security services are self-encapsulated software modules which deliver service functionalities through standard Internet communication protocols. Their implementation details are hided from service requesters, thus any internal system modification will not affect the requester side. Besides, the underlying system complexity does not impede other systems to consume those services, because they are accessible only through high level interface.

2. **Standardized protocols** – Web Service protocol stack comprises four layers: service transport layer, XML messaging layer, service description layer, and service discovery layer. This gives the possibility to choose from broad collection of well defined standard protocols for particular system implantation.

3. **Interoperability** – the most significant advantage that the security system benefits from the Web Service technology is interoperability. The system delivers its services through public network, such as Internet, in an interoperable manner independent of its implementation platform. Platform-independent service provider and requester communicate with each other without any obstacles.

4. **Usability** – Web Service are used by client applications. Regardless what tools and programming languages are used for its implementation, the client application connects to the service end-point and makes service calls on that endpoint. The communication takes place by means of serialized request-response messages using standard data formats, like XML.
5. **Deployability** – security services are deployed over standard Internet technologies on application servers and all incoming and outgoing messages can easily pass through firewalls, using SSL over HTTP channel security mechanism.

Web Service technology facilitates managing security solutions for our system through different built-in security mechanisms provided by Web Services Security protocol. All these advantages make the system less costly to implement and deploy.

The only disadvantage that the Web Service technology may cause to the security system is the stateless interaction between the service requester and provider. That is why additional mechanisms may be required in order to keep track of service requests.

### 6.2. Evaluation of System Security

Security evaluation is based on the attack-oriented threat model. Threat model gives a formal approach to order potential security issues that makes the system security evaluation easy to understand. The proposed security system is analyzed for possible security threats, taking into account security considerations for both authentication and authorization systems, highlighted in Chapter 3. There are five defined possible attacks for both services: replay attacks, message information disclosure (confidentiality), message modification, impersonation, and repudiation. Table 1 shows whether both services are protected against those security threats. Replay attacks can be prevented using randomly generated session IDs (assertion IDs) in messages for both services. Message confidentiality is also protected for both services: in case of authentication service, messages are transmitted over a secure channel, such as SSL/TLS and in case of authorization service, messages can be encrypted using XML encryption standard. Message modification and repudiation are prevented using XML digital signature standard for both services. Impersonation attack is also prevented for both services using XML digital signature standard, as it can also provide information that the message is originated from intended entity.
Table 1: System Security Evaluation

<table>
<thead>
<tr>
<th>Security Services</th>
<th>Replay Attacks</th>
<th>Message Information Disclosure</th>
<th>Message Modification (Tampering)</th>
<th>Impersonation</th>
<th>Repudiation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Authorization</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

6.3. Summary
This chapter has presented the evaluation of the proposed security services from two aspects: integration and security. All advantages and disadvantages adopted from the Web Service technology have been highlighted for our system. Finally, it has been shown that the system is resistant to all potential security issues associated with channel level for both services.
7. Conclusions and Future Work

This chapter summarizes the overall investigation of this thesis and recommends some future work in the research area.

7.1. Conclusions

Implementations of cloud security solutions under the concept of Security as a Service are in their awaking phase. This research has proposed a cloud security system based on that concept and made contributions in the area of authentication and authorization services for a cloud environment. The problem has been solved and the goals have been achieved.

The architecture of the proposed cloud security system consists of two components: central security servers and portal security servers. Central security servers are responsible to provide two identity services, such as authentication and authorization services for cloud-based software services. Both identity services were designed using Web Service technology and XML-based standards. Portal security servers are responsible to protect cloud portals based on the services delivered from central security servers. Portal security server takes the role of a proxy server, which provides Policy Enforcement Point services to application services of a cloud portal. In this way, authentication and authorization solutions are decoupled from individual application services and delegated to the shared cloud security system, which deliver these identity services through SecaaS model.

Centralization and sharing of those identity services in a separate security infrastructure results in an effective and flexible solution for a cloud environment. This approach enables the entire cloud security system to be controlled and managed much easier, thus raising the quality of provided cloud security solutions. Besides, the system ensures the provisioning of those identity services in a secure and reliable manner.

A prototype of authorization service has been implemented in order to demonstrate the possible application of the designed cloud authorization system. It is a role-based authorization system with minimal necessary features. This prototype implementation consists of two parts: authorization service and access control administration. The authorization service is implemented using Web Service technology and access control administration is implemented with simplified functionality as a user friendly web-based application.

Through this research a solution is provided for building cloud-based identity services, such as authentication and authorization based on the cloud SecaaS model. This solution aims to provide an open and platform-independent architecture of a cloud security
system, which is completely service-oriented, thus enabling the system to be scalable, interoperable, loosely coupled and location transparent.

7.2. Future Work
In this research a cloud security system has been designed for managing authentication and authorization services applying quite new cloud service paradigm, such as Security as a Service. As such, there is a need to do more comprehensive observations and activities within this area and here are some of them:

- Cloud-based security service providers deal with end-users whose privacy should not be violated at all. Although the system promises that from theoretical perspective according to the applied security techniques and approaches, there is a need to conduct focused practical activities within this area in order to see the real picture of the security system robustness against potential privacy vulnerabilities. At the same time all security credentials are stored in the central security system, which makes it possible to link and trace end-user activities by cloud identity service provider.

- Centralization of the identity services for a cloud environment represents another two issues: single point of failure and single target of attack. Therefore, there is a need to conduct extra work related to data replication and protection for solving those mentioned problems.

- Security evaluation of the proposed security services is based on security considerations associated only with communication channel level security risks. That is why there is a necessity to make additional security evaluations against security issues associated with other system aspects, such as hardware, software, etc.

- System performance should be evaluated in a scalable environment in order to measure how responsive it is in case of large amount of service requests. This will also show how resistant the system is against denial of service attacks.

- The proposed system supports delivery of only two identity services. Therefore, more identity service features can be added, such as single log out, session refreshment, etc.

- The prototype implementation has some limitations: user can be assigned only one role at a time, there is no policy set concept applied for this system, and there is no separately implemented Policy Information Point service. Therefore, more features can be added to the prototype authorization system. Besides, a prototype of authentication system can be implemented according to the designed system.
Bibliography


Appendix A

The WSDL file for the PDP Web Service Interface:

<!--
Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is Metro/2.2.1 (tags/2.2.1-7242; 2012-08-03T12:35:22+0000) JAXWS-RI/2.2.7 JAXWS/2.2 svn-revision#unknown.
-->

<!--
Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is Metro/2.2.1 (tags/2.2.1-7242; 2012-08-03T12:35:22+0000) JAXWS-RI/2.2.7 JAXWS/2.2 svn-revision#unknown.
-->

    <sp:AsymmetricBinding>
      <wsp:Policy>
        <sp:AlgorithmSuite>
          <wsp:Policy>
            <sp:Basic128/>
          </wsp:Policy>
        </sp:AlgorithmSuite>
        <sp:IncludeTimestamp/>
        <sp:InitiatorToken>
          <wsp:Policy>
            <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient"/>
          </wsp:Policy>
        </sp:InitiatorToken>
      </wsp:Policy>
    </sp:AsymmetricBinding>
  </wsp:Policy>
</definitions>
</wsp:Policy>
</sp:InitiatorToken>

<sp:Layout>
<wsp:Policy>
<sp:Strict/>
</wsp:Policy>
</sp:Layout>

<sp:OnlySignEntireHeadersAndBody/>

<sp:RecipientToken>

<wsp:Policy>
<sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">

<wsp:Policy>
<sp:RequireIssuerSerialReference/>

<sp:WssX509V3Token10/>
</wsp:Policy>
</sp:X509Token>
</wsp:Policy>
</sp:RecipientToken>
</wsp:Policy>
</sp:AsymmetricBinding>

<sp:Wss10>

<wsp:Policy>

<sp:MustSupportRefIssuerSerial/>
</wsp:Policy>
</sp:Wss10>

<wsam:Addressing/>
</wsp:Policy>


<sp:EncryptedParts>
</sp:EncryptedParts>

<sp:Body/>
</sp:EncryptedParts>
<sp:SignedParts>
<sp:Body/>
<sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
<sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
<sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
<sp:Header Name="AckRequested" Namespace="http://docs.oasis-open.org/ws-rx/wsrm/200702"/>
<sp:Header Name="CreateSequence" Namespace="http://docs.oasis-open.org/ws-rx/wsrm/200702"/>
<sp:Header Name="Sequence" Namespace="http://docs.oasis-open.org/ws-rx/wsrm/200702"/>
<sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
<sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
<sp:Header Name="SequenceAcknowledgement" Namespace="http://docs.oasis-open.org/ws-rx/wsrm/200702"/>
<sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
<sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
</sp:SignedParts>
</wsp:Policy>
<sp:EncryptedParts>
<sp:Body/>
</sp:EncryptedParts>
<sp:SignedParts>
<sp:Body/>
</sp:SignedParts>
</wsp:Policy>
<sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
<sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
<sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
<sp:Header Name="AckRequested" Namespace="http://docs.oasis-open.org/ws-rx/wsrm/200702"/>
<sp:Header Name="CreateSequence" Namespace="http://docs.oasis-open.org/ws-rx/wsrm/200702"/>
<sp:Header Name="Sequence" Namespace="http://docs.oasis-open.org/ws-rx/wsrm/200702"/>
<sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
<sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
<sp:Header Namespace="http://docs.oasis-open.org/wsrpwsrm/200702" Name="SequenceAcknowledgement"/>

<sp:Header Namespace="http://www.w3.org/2005/08/addressing" Name="Action"/>
<sp:Header Namespace="http://www.w3.org/2005/08/addressing" Name="RelatesTo"/>
</sp:SignedParts>
</wsp:Policy>
<types>
<xsd:schema>
</xsd:schema>
<xsd:schema>
</xsd:schema>
<xsd:schema>
</xsd:schema>
</types>
<message name="requestXACML_Authorization_Decision">
<part name="parameters" element="tns:requestXACML_Authorization_Decision"/>
</message>
<message name="requestXACML_Authorization_DecisionResponse">
<part name="parameters" element="tns:requestXACML_Authorization_DecisionResponse"/>
</message>
<message name="IOException">
<part name="fault" element="tns:IOException"/>
</message>
<portType name="PDPServiceProvider">
<operation name="requestXACML_Authorization_Decision">
"
<fault message="tns:IOException" name="IOException" wsam:Action="http://authorization_service/PDPServiceProvider/requestXACML_Authorization_Decision/Fault/IOException"/>
</operation>
</portType>
(binding name="PDPServiceProviderPortBinding" type="tns:PDPServiceProvider">
<wsp:PolicyReference URI="#PDPServiceProviderPortBindingPolicy"/>
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document">
<operation name="requestXACML_Authorization_Decision">
<soap:operation soapAction=""/>
<input>
<wsp:PolicyReference URI="#PDPServiceProviderPortBinding_requestXACML_Authorization_Decision_Request_Policy"/>
<soap:body use="literal"/>
</input>
<output>
<wsp:PolicyReference URI="#PDPServiceProviderPortBinding_requestXACML_Authorization_Decision_Response_Policy"/>
<soap:body use="literal"/>
</output>
<fault name="IOException">
<soap:fault name="IOException" use="literal"/>
</fault>
</operation>
</binding>
<service name="PDPServiceProvider">
<port name="PDPServiceProviderPort" binding="tns:PDPServiceProviderPortBinding">
<soap:address location="http://130.237.215.216:8080/PDPService/PDPServiceProvider"/>
</port>
</service>
</definitions>
Appendix B

A sample of policy file:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" PolicyId="oooooo"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides"
Version="2.0" xsi:schemaLocation="http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-os.xsd">

<Description> this a policy for manager role </Description>

<Target>

<Resources>

<Resource>

<ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">

<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">http://130.237.20.77:8080/SecureCloudEmail/</AttributeValue>

</ResourceMatch>

</Resource>

</Resources>

</Target>

<Rule Effect="Permit" RuleId="rule1">

<Target>

<Actions>

<Action>

<ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">

<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>

<ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false"/>

</ActionMatch>

</Action>

</Actions>

</Target>

<Condition>

</Condition>

</Rule>

</Policy>
```