Policy enforcement in cloud computing

Gurbanov, M.

Award date:
2013

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science

Policy Enforcement in Cloud Computing

For the degree of Master of Science in Information Security Technology

Murad Gurbanov

Examination Committee:
Dr. Milan Petkovic (*Supervisor*)
Paul Koster (*On site supervisor*)
Tanir Ozcelebi

Eindhoven, October 2013

"Research is what I'm doing when I don't know what I'm doing."
Wernher von Braun

Abstract

Cloud Computing is an emerging technology, providing attractive way of hosting and delivering services over the Internet. Many organizations and individuals are utilizing Cloud services to share information and collaborate with partners. However, Cloud provides abstraction over the underlying physical infrastructure to the customers, that raises information security concerns, while storing data in a virtualized environment without having physical access to it. Additionally, certain standards have been issued to provide interoperability between users and various distributed systems(including Cloud infrastructures), in a standardized way. However, implementation and interoperability issues still exist and introduce new challenges.

This thesis explores the feasibility of securing data in a cloud context, using existing standards and specifications, while retaining the benefits of the Cloud. The thesis provides a view on increasing security concerns of moving to the cloud and sharing data over it.

First, we define security and privacy requirements for the data stored in the Cloud. Based on these requirements, we propose the requirements for an access control system in the Cloud. Furthermore, we evaluate the existing work in the area of currently available access control systems and mechanisms for secure data sharing over the Cloud, mostly focusing on policy enforcement and access control characteristics. Moreover, we determine existing mechanisms and standards to implement secure data sharing and collaborative systems over the Cloud.

We propose an architecture supporting secure data sharing over the untrusted Cloud environment, based on our findings. The architecture ensures policy based access control inside and outside Cloud, while allowing the benefits of Cloud Computing to be utilized. We discuss the components involved in the architecture and their design considerations.

To validate the proposed architecture, we construct the proof of concept prototype. We present a novel approach for implementing policy based access control, by achieving interoperability between existing standards and addressing certain issues, while constructing the system prototype.

Furthermore, we deploy our solution in the Cloud and perform the performance tests to evaluate the performance of the system. Finally, we perform a case study by utilizing our system in a real-life scenario. To do this we slightly tailor our solution to meet specific needs.

Overall, this thesis provides a solid foundation for the policy enforcement and access control mechanisms in the Cloud-based systems and motivates further work within this field.

Acknowledgments

I would like to offer my sincere gratitude to Dr. Milan Petkovic and Paul Koster who have guided me through every aspect of this thesis work. I am grateful for their motivation, knowledge and time for long disucssions, which helped me invaluably in solving the problems addressed in this thesis. It was a wonderful learning experience that had given me knowledge and skills required for researching and solving complex problems.

I would like to thank Muhammad Asim for his valuable suggestions and time during the course of this thesis. The suggestions helped me into looking at the problem from different perspectives and thinking about better solutions.

I finish my Kerckhoffs masters program at TU/e with this thesis. I am grateful to TU/e for providing me with a great study environment and research facilities. I am thankful to all my teachers to providing me with invaluable knowledge and encouragement while my study period.I have gained immense knowledge in various aspects of information security and life in general during my stay in Eindhoven.

Finally, I want to thank my parents, my beloved fiancee, and all of my friends, for their continued support during the project. I highly appreciate their help and motivation, that keep me focused over the entire period of working on this thesis.

Contents

Contents **v**

List of Figures **viii**

List of Tables **ix**

1 Introduction **1**

2 Secure Data Access in Cloud **4**
2.1 Cloud Computing 4
2.2 Security and Privacy Requirements in Cloud Computing 6
2.3 Access Control 9
2.4 Access Control Requirements in Cloud 10

3 Related Work **13**
3.1 Secure Data Storage in Cloud 13
3.2 Access Control Systems in Cloud 14
3.3 Policy Based Access Control in Cloud 16
3.3.1 Data Centric Policies 17

4 Preliminaries **19**
4.1 Rest Architecture 19
4.2 Attribute Based Access Control 20
4.3 Cloud Data Management Interface 20
4.4 Key Management Interoperability Protocol 21
4.5 eXtensible Access Control Markup Language 22

5 System Architecture **24**
5.1 Architecture 24
5.1.1 Users 25
5.1.2 Cloud-client application 25
5.1.3 Data Management Server 26
5.1.4 Key Management Server 26
5.2 Authentication 27
5.3 Authorization 27
5.4 Key management 28
5.5 Encryption 28

5.6 Data Operations . . . 29
5.6.1 Data Upload . . . 29
5.6.2 Data Download . . . 30
5.6.3 Data Operations Outside Cloud . . . 31
5.6.4 Additional Notes . . . 32

6 Implementation **34**
6.1 Implementation Structure . . . 34
6.2 Authentication . . . 35
6.3 Policy Enforcement . . . 35
6.3.1 PDP . . . 35
6.3.2 PEP . . . 36
6.3.3 Authorization Proxy . . . 36
6.4 Data Management . . . 37
6.4.1 Data Upload . . . 37
6.4.2 Data Download . . . 39
6.4.3 Data Sharing . . . 40
6.5 Key Management . . . 40
6.5.1 KMIP Parser . . . 40
6.5.2 Key Registration and Retrieval . . . 40

7 Evaluation **42**
7.1 Methodology . . . 42
7.2 Environment . . . 43
7.3 Results . . . 43
7.3.1 Data Upload . . . 44
7.3.2 Data Download . . . 48
7.3.3 Data Sharing . . . 54
7.4 Discussions . . . 55

8 Conclusions and Future work **56**
8.1 Conclusions . . . 56
8.2 Future Work . . . 57

A Case Study: Emergency Medical Service **59**
A.1 Use Case . . . 59

B Available CDMI Servers **61**

C KMIP Messages **62**
C.1 Key registration request. . . . 62
C.2 Key registration response. . . . 63
C.3 Key retrieval request. . . . 63
C.4 Key retrieval response. . . . 64

D XACML examples **66**
D.1 Server-side policy example. . . . 66
D.2 Data-centric policy example. . . . 67

Bibliography **69**

List of Figures

2.1 The NIST definition of Cloud Computing. 5
2.2 Core components of access control system. 10

3.1 Data-centric security model. 17

5.1 Architecture of the cloud data sharing system. 24
5.3 Data download process. 31
5.4 Data sharing process. 32
5.2 Data upload process. 33

6.1 Implementation architecture of the cloud data sharing system. 34

7.1 Average data upload time to the DMS. 44
7.2 Average AES encryption performance on m1.medium instance.. 45
7.3 Average Base64 encoding performance on m1.medium instance.. 45
7.4 Average performance of the complete file upload process on m1.medium instance. 46
7.5 Performance of the complete file upload process on m1.medium instance. . . 46
7.6 Sequence of operations while data upload. 47
7.7 Relative timing of operations not depending from file size while data upload. 49
7.8 Average data download time to the DMS. 49
7.9 Average AES decryption performance on m1.medium instance.. 50
7.10 AES decryption performance on m1.medium instance.. 51
7.11 Average Base64 decoding performance on m1.medium instance.. 51
7.12 Average performance of the complete file download process on m1.medium instance.. 52
7.13 Performance of the complete file download process on m1.medium instance. 52
7.14 Sequence of operations while data download. 53
7.15 Relative timing of operations not depending from file size while data download. 54
7.16 Performance of policy retrieval from DMS. 54
7.17 Performance of policy evaluation on local machine (TUA). 55

A.1 Use case scenario for an emergency medical service. 59

List of Tables

3.1 Comparison of access control systems on cloud. 16

7.1 Configuration of the m1.medium instance in Amazon EC2. 43
7.2 Execution time of the tasks, while data upload, not depending from the file size. 48
7.3 Execution time of the tasks, while data download, not depending from the file size. 53

B.1 Comparison of CDMI implementations. 61

Chapter 1

Introduction

Advanced technologies offer new opportunities to the industry. Cloud computing is one of these technologies. It provides computing power and storage resources in distributed environment, while abstracting underlying infrastructure, depending on a given service.

Several enterprise level organizations deploy highly scalable cloud computing solutions for an internal data sharing and collaboration. Moreover, some companies offer their cloud services for commercial use and act as a Cloud Service Providers (CSP) in the market.

CSPs claim to provide better security, reliability, sustainability, cost effectiveness and support than IT systems of individual organizations. These features make it possible to move business from individual systems to the cloud and make it accessible over the Internet. Dynamic nature, high scalability and extensive computing resources make a cloud environment ideal for collaborative research and data sharing.

All aforementioned benefits of cloud computing results in a higher attention to this technology, and highlights a significant importance of the cloud, with regards to how information shared and processed in a modern society. Therefore, we assume that usage and importance of cloudbased solutions will increase significantly in the time coming.

However, in some cases collaborative research and data sharing involves processing and storage of sensitive data, which is considered as private information and should not be shared without consent(for example patient consent in healthcare).

While cloud computing is emerging rapidly and utilized by increasing number of organizations worldwide, data confidentiality and integrity issues are not sufficiently addressed currently. Storing sensitive data in the cloud, without knowledge where data is really resided and making this data accessible over the Internet increases the risk of data compromise. The risk of confidential data leakage and privacy violations in the cloud significantly hinders a wide adoption of this technology [67].

CSPs offer various measures to protect the data stored in the cloud. Most of the CSPs offer encryption capabilities to the customers, so that all data is transferred and stored in an encrypted form in a cloud storage system. However, key management challenges and insider threats are still should be addressed. Certification of CPSs may provide some level of assurance to the customers. However, there are still no guarantees of full self control over the data resided in the cloud, for the customers.

Additionally, the lack of standardization, with regard to access cloud services, decreases interoperability and flexibility of switching among CSPs . Hence, organizations may experience vendor lock-in, unless they are not willing to put a significant effort to accommodate

their existing solutions to the new CSP.

Therefore, careful system design, review of existing standards, risk management and requirements analysis is needed before deploying services in the cloud. Access control systems should be reviewed and tailored for the suitability with the dynamic and distributed nature of the cloud environment. Wide variety of the use cases and imperfections of still developing cloud technologies and standards makes the aforementioned tasks even more complex and hard to realize in a unified way.

In this thesis, we address the following research questions:

- What are the requirements for a secure data access in a cloud environment?
- Is it possible to design a cloud data management system, utilizing the benefits of cloud storage, while ensuring data security and fine-grained data access.
- Feasibility of the development and deployment of such systems in the cloud, based on existing standards and technologies.
- How much is the performance impact for such systems while operating in a cloud environment?

To answer these questions, we introduce the following contributions:

- Research on existing work in the area of designing secure data sharing systems in the cloud. Research involves analysis of existing literature and already existing implementations.
- Defining requirements for access control system in cloud environment. We first define security requirements for the data resided in cloud. Then, based on these requirements, we define considerations for access control system in cloud.
- Designing a secure data management system for a cloud environment. In particular, we propose the architecture of the system utilizing policy based access control to the data.
- Implementation of the proposed architecture in the cloud, based on existing standards and technologies. First, we research for the existing standards and technologies, and their potential adoption for the cloud environment. Later, we implement a prototype of the proposed architecture, based on our findings.
- Performance analysis of the developed prototype. We deploy our prototype in a cloud environment and conduct tests to measure its performance under various scenarios.

The structure of the thesis is as follows.

Chapter 2 introduces the cloud computing concepts and security requirements for data sharing in cloud.

Chapter 3 explores the related work and existing systems for a secure data sharing in cloud.

Chapter 4 explains the preliminaries required for the understanding of the following chapters.

Chapter 5 proposes the architecture of the system, and discusses various aspects of it.

Chapter 6 explains the implementation of the proposed architecture, based on chosen standards and technologies.

Chapter 7 provides evaluation of system from performance and security point of view.

Chapter 8 presents the conclusions on the performed work and future work in this area.

Chapter 2

Secure Data Access in Cloud

In this chapter we introduce cloud computing concepts and its characteristics. We explain benefits provided by this technology, followed by the drawbacks and security concerns while using this technology. Then, we specify number of requirements to ensure data security in the cloud. Furthermore, briefly explain the role and basic principles of access control. Finally, we specify requirements for an access control systems to support secure data storage and collaboration in the cloud.

2.1 Cloud Computing

Cloud computing is an evolving paradigm, that involves development of related technologies, to provide extensive computing power, storage, high availability and relatively low cost.

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." according to the NIST definition of Cloud Computing [39].

Cloud computing provides abstraction from underlying infrastructure and mechanisms, for applications and services. NIST proposes cloud model as described in Figure 2.1

Typically, cloud model exhibits five characteristics:

- **On-demand self-service.** Computing resources are available for the users on demand and without any intermediate parties.Computing power, network storage, memory, server time, etc. may be an example for the resources, in the current context.
- **Broad network access.** Resources can be accessed over the network, using predefined mechanisms. Typically, mechanism are determined by the CSPs. Cloud services are accessible by various client platforms, such as laptops, mobile phones, tablets, workstations, etc.
- **Resource pooling.** Resources of CSP are pooled to provide service to multiple consumers, using multi-tenant model, with dynamic allocation of physical and virtual resources, depending on the consumer needs. Customers are not aware about exact

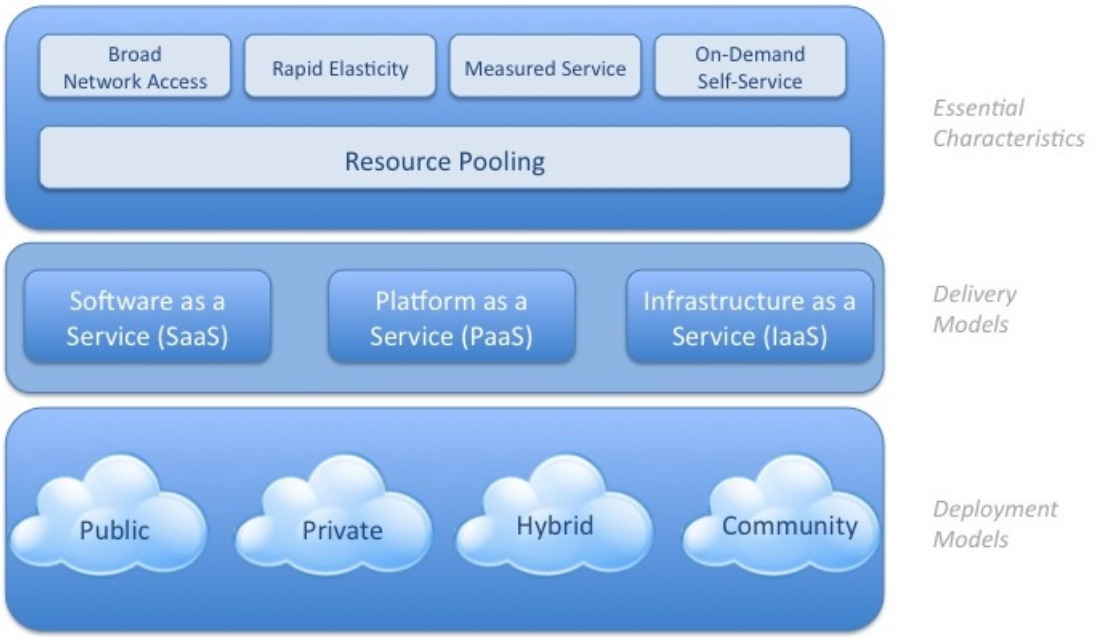

Figure 2.1: The NIST definition of Cloud Computing.

location of the provided resources, since cloud provides high availability from any point of access. However, customers may specify location of the resources at a higher level of abstraction.

- **Rapid elasticity.** Cloud resources appear to be unlimited to the consumers. However, resources can automatically scale up or down commensurate with demand. Users may also control the usage of the resources manually, and increase or decrease resource capacity on demand.
- **Measured service.** CSPs automatically optimize and control resources, according to the type of service. This is typically achieved by ”pay per use” and ”charge per use” principle, according to NIST. User activity and resource usage can be monitored, controlled and reported transparently for both, consumer and service provider.

There are three cloud service models specified by NIST.

- **Software as a Service (SaaS).** This type of service provides highest level of abstraction to the users. Users are provided with the set of applications deployed on a cloud infrastructure, by CSP. Typically, the applications are accessible over the Internet, through a web browser interface or program interface. Users does not control accessible resources of the underlying cloud infrastructure, such as storage, network, servers, etc. Instead, cloud applications are responsible for automatic scaling of the resources.
- **Platform as a Service (PaaS).**Users are provided with the capability to deploy their own applications onto the cloud, using predefined set of operating systems and toolkits. Additionally, users may configure the settings of the application-hosting environment. However, resources of underlying cloud infrastructure are still controlled and managed by the CSP.

- **Infrastructure as a Service (IaaS).** This type of service provides lowest level of abstraction to the users. Users are provided with the raw storage space, computing and network resources, where they can deploy and run arbitrary software, including operating systems and applications. Underlying cloud infrastructure is still managed by CSP, but users have full control over operating systems, storage, applications and limited control over certain network resources.

NIST classifies four deployment models, based on the underlying infrastructure.

- **Private cloud.** Clouds implementing this deployment model are intended to be used by a single organization. It is owned and managed by organization itself, or a third party responsible to provision of private cloud services. Generally, private clouds offer better quality of service. However, they require significant investment and management effort.

- **Community cloud.** Some organizations may have shared concerns and similar requirements. These organizations may benefit from deploying shared cloud environment, that is managed by one or more attending organizations, a third party, or their combination. Community cloud may be a good choice for the organizations with moderate finances and requiring partially managable cloud services, at lower cost.

- **Public cloud.** This type of cloud environments are provided to the general public by a CSPs. Multiple customers may share the public cloud, which results in lower service costs for the individual consumer. However, customers have less control over the cloud infrastructure in comparison with private and community clouds. Additionally, data is shared with unknown parties in a physical level. This consideration should be taken into account, before storing sensitive data in public cloud.

- **Hybrid cloud.** Hybrid clouds are the composition of multiple cloud infrastructures bound together by certain technology, enabling interoperability between cloud services. Hybrid clouds can consist of private, public and community cloud infrastructures.

Public cloud is a most demanding deployment model, since it provides most significant and beneficial features of cloud, such as lower costs, extensive computing power, rapid scalability, etc. However, there are number of concerns and arising requirements while moving business to the public cloud[1].

2.2 Security and Privacy Requirements in Cloud Computing

Protection of data is an important requirement while designing IT infrastructure of the organization. These requirements get even more stringent when data moves to cloud, and data becomes accessible through Internet. Therefore, strong authentication and access control mechanisms should be in place while deploying such systems.

Below we will describe general security and privacy requirements for the data resided in cloud, based on review of the related literature and our own analysis.

[1] We will refer to the "public cloud" as a "cloud" throughout this thesis for visual comfort.

Authenticity of Data

Data stored in cloud should be authentic, which means it should be possible to determine if the data is genuine and to verify the creator or owner of the data [65]. Deceptive data might result in unpredictable consequences, depending on use case scenario. Therefore, it should be possible to determine the owner of the data for further investigation. Note that owner or creator of the data may be a group or certain set of users.

Authenticity of the data might be achieved by various signature schemes[59].

Authentication

Authentication of users and components is a crucial part of the collaboration in cloud. Data should be accessed only by the legitimate users. Therefore, all users are required to perform authentication process and getting authorized by the system, before allowing them to perform any other operations. Unauthorized access to the data might result in information leakages, data manipulation and other undesirable consequences.

Non-repudiation

Changing or deleting any valuable data might have significant consequences and result in money and reputation losses. Collaboration in cloud may deal with data provided by various data providers, and inconsistencies in these data might lead to reputation and money losses of data providers. Inconsistencies might occur by the ill-intentioned actions of the users and other involved parties.

Non-repudiation property ensures that users cannot deny their transactions and actions on datasets[23], and will carry responsibility over them.

Integrity

Collaboration in cloud may require large amount of data to be processed. Therefore, accuracy and consistency of processed data as well as obtained results should be preserved[65]. Inconsistencies of the stored data might result in unpredictable consequences, depending on use case. For example, integrity violations may result in large amount of re-computations, and in wrong outcomes, which is unacceptable risk in some cases(e.g. healthcare systems).

Confidentiality

Unauthorized insiders or systems should not have permissions and ability to access data, which may considered as an internal documents, trade secrets, intellectual property, etc. In the cloud computing environment data is distributed over the remote servers, that are shared with others and can be accessed through Internet or other connections.

Moreover, collaboration might involve multiple cloud providers into business[50]. These factors will increase the threat of data compromise in the cloud, since increased or even unknown number of access points will be available to the sensitive data[40]. Therefore, serious confidentiality considerations should be taken into account, before moving data into the cloud.

It is possible to achieve confidentiality through encryption mechanisms[65]. However, in most of the cases data should still be searchable[32] in order to process queries on datasets

and to support certain activities. Techniques like searchable encryption or maintaining index of data might help in this case[7].

Additionally confidentiality should be preserved while transporting the data[24], therefore encrypted channels like TLS, VPNs, etc. should be considered[38].

Availability

Users of a cloud should be able to access and use needed data resources on demand. High availability should be achieved, by preventing situations, such as Denial-of-Service attacks, power outages and hardware failures[18].

Since cloud provides extensive processing power, organizations may perform their data analysis more faster and effectively using cloud resources. However, analysis of data might suffer from unavailability of the data resources needed for analysis on demand.

Auditability

Audit procedures should be in place, in order to maintaining log of every access and modification of data in distributed environment[23]. Security breach notifications might be generated automatically based on logs.

Audit logs might be stored and managed decentrally, however, it should be possible to reconstruct prior state of information and replay performed actions by combining relevant logs together. Audit logs might be the only way to track changes in the datasets.

Additionally, confidentiality, integrity and availability of audit logs must be ensured.

Backup Procedures

All data stored in the cloud system should be backuped, in order to prevent data losses. Cloud service providers establish backup procedures to their costumers as an indistinguishable part of their service. However, it might be risky to allow the cloud service providers to back up sensitive data and store it in distributed environment[23]. Even if built-in backup procedure is one of the benefits of cloud service, it might have a negative effect when sensitive data is considered.

Additionally, backups may raise a concern about deletion of the data. How to deal with the situations, when sensitive data should be deleted, but it was backuped? Deleting backup copy is not a solution, as there might be multiple backups and in real-life scenario backups may contain combined data, which will also be deleted in that case. Backup strategy for data resided in cloud must be negotiated with cloud provider according to the policies and the needs of parties involved.

Data Location Restrictions

Data in the cloud is stored in the distributed manner, in the servers located all over the world. There is a need to restrict locations where cloud service provider will physically host the data[45], before storing data or deploying corporative systems in the cloud. It is reasonable to avoid countries where intellectual property and privacy laws are inadequate and where data may be a subject of investigation due to the intrusive nature of the government.

Data Traceability and Labeling

All data stored in the cloud must be traceable in order to identify the origin of the leakage. Additionally, confidential data should be labeled, indicating that the data is proprietary and unauthorized usage of it will have legal consequences. These requirements are especially important in cloud environment, where increased vectors of information leakage present.

Data Segmentation

Cloud computing benefits from resource sharing and virtualization mechanisms. That means multiple virtual machines are running on one physical machine and controlled by software hypervisor to keep appropriate separation of resources[45]. Cloud provides less separation than private IT infrastructures, where separate physical servers are deployed. So, there is a potential threat of data compromise through virtual machines running under others control on the same physical server where sensitive data of certain organizations might be stored. Also, there is a risk of the network traffic capture in such cases.

In ideal, data of separate parties should be processed and stored through unshared physical machines[24], which is not the conceptional idea of a cloud computing. Therefore resource sharing must be limited to the minimum, depending on the agreement with the cloud provider.

It is worth to note that, due to the resource sharing between different organizations in the cloud, forensic inspections of the storage will be a legal challenge.

Aforementioned requirements should not be underestimated when moving to a cloud service, even if some of them are not feasible yet. Literally, control over the data is lost after the data has been moved to a cloud provider. It is not clear where data is resided and if data is disclosed to unauthorized parties. Moreover, it is not always clear who is the owner of the data, if data moved or created in the cloud.

2.3 Access Control

*"Access control is the process of mediating every request to resources and data maintained by a system and determining whether the request should be granted or denied."*according to the definition given by Samarati and Vimercati [48].

Access control is an important part of any information system. Properly implemented access control should guarantee "secrecy" and "integrity" of the data and resources, against unauthorized disclosure and modifications. Additionally, access control should provide legitimate users with "availability", to be able access the data and resources on demand.

Access control systems are implemented based on rules, according to which access will be controlled. Particularly, access is controlled by "authorization". Basically, authorization is a permission to access a requested resource.

Implementation of access control systems usually based on three concepts:

- **Security policy** is a set of rules governing access control decisions in the system.
- **Security model** is a formal representation of the access control system and security policy, proofing the security properties provided by access control system.

- **Security mechanism** is a functions that implement the security model and security policies. Functions can be implemented by both, software and hardware.

Access control systems consist of certain core components, providing basic functionality. Figure 1 represents a basic access control architecture containing core components.

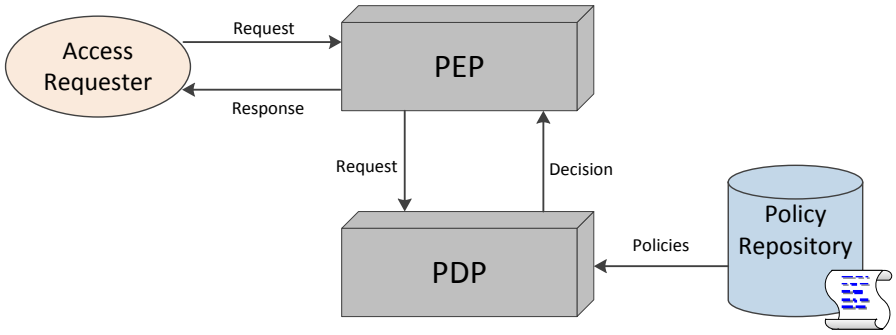

Figure 2.2: Core components of access control system.

An access requester makes an authorization request to the policy enforcement point (PEP) to perform certain actions on the resource. PEP forwards authorization request to the policy decision point (PDP), that evaluates request against existing policies. Policies are stored in the policy repository. PDP responses the decision to the PEP. PEP enforces the decision and provides the response to the requester.

2.4 Access Control Requirements in Cloud

Specifying all access control requirements is an inherently difficult task due to the extensive number of use cases. Below we will describe requirements for an access control mechanisms for a secure data storage and collaboration in the cloud, taking into consideration general security requirements described before (Section 2.2).

Granular Access Control

Access control system for cloud should support a wide range of resource types, to support a collaboration of various customers.

Basic operations, such as read, write, execute, delete should be supported. Additionally fine-grained access methods might be supported, depending on specific needs of performed operations.

Defining a purpose of an operation might be strictly required in some cases, therefore access control system should support this feature. For example, data might be accessed for a research activities, but not editing.

Decentralized Access Control

Collaboration in cloud may collaborate multiple ICT infrastructures of different participants and centralized access control approach will not fulfill the requirements of this ecosystem[37].

Most of the current systems apply role based access control, which is suitable for homogeneous environments, where similar nature of data, roles and tasks allows to do so. However, in systems deployed in cloud and spanning multiple countries, support for a more diverse objects, users and rules must be considered[7].

Most of the organizations have their own identity management systems, specific roles and policies deployed organization-wide. Therefore, it is desirable for participating organizations to retain their internal organizational structure even after joining the collaboration. For example, an organization might have policy, that restricts connections to any external data sources while working with organization's internal data source, in order to cope with the risks of data leakage. Employee has a role with specific access privileges to the internal data source of the organization, however this access privileges are not supported by the system in cloud, where employee also considered as a consumer. Therefore, local administrators should configure access to the system for that employee from the internal workspace, so that employee can perform permitted operations. Administrators should follow particular procedures, defined by the organizational policies. Moreover, organizations access control system might need a purpose to be specified in the request, whereas system in the cloud might not support this feature.

Obviously, it is unrealistic to impose single access control mechanisms for all participants due to the technological and legal challenges.

Overall, major requirement for an access control system in cloud is to allow decentralized, autonomous access control[53].

Access Control Negotiation

Different legal and ethical policies of participating organizations bring number of challenges for an access control in cloud. Request for data might be accepted or rejected depending on law enforcements of the country where data resides. In some countries ethical approval might be needed in order to fulfill the request[53]. Access control system should support such constraints, scenarios and provide additional information on replies to the requester, in order to modify request and negotiate the appropriate type of access.

Furthermore, access control system should be able to negotiate access automatically or interactively.

Ownership and Delegation

All data in the cloud should have an owner. Owner of the data can be an organization, group or user. Establishing the owner of the data is necessary to prevent unauthorized access to protected data, as an owner can define rules of access to the information. Owners have full permissions on their data and can regulate access to it.

Additionally, users and owners should be able to delegate their permissions or subset of their permissions to other users in distributed environment. This will offload burden of access rights management from the owners of the data and the administrators of an organization.

It is worth to note that, depending on system design, data might have multiple owners in a hierarchial manner[7]. For example, both, organization acting as a data provider and its employee might be the owner of the data. Regulatory policies of the organization will determine who will entitled for higher privileges. For example, researchers executing the

analysis based on the provided data might be considered as the owners of the results of that analysis. However, depending on the system design, researchers might have full or shared ownership with data provider, on the results.

Least Privilege

Users should be given only required permissions to perform their tasks, and restricted from accessing data outside of their domain. This will lead to the limited audience access for a particular set of data, which will decrease information leakage paths[7].

Additionally, in most real world scenarios, administrators are given excessive privileges, which is a significant risk to a data confidentiality. There should be mechanism in place, to allow administrators to administer data, without revealing data itself[45][33].

Spatio Temporal Constraints

Data resided in cloud might be accessed from any location at any time. Access control system should be able to regulate access to the data considering location and time constraints[32]. This should be done to enable access to the data only from authorized locations and in certain time periods. For example, only employees of particular organization can access certain data, from their working places, in a given time period.

Since cloud providers offer services to various customers, access control requirements may vary from case to case, making implementation of common access control system even more challenging.

Chapter 3

Related Work

In this chapter, we analyze the existing work in the areas of secure data storage and access control in a cloud environment.We present the evaluation of access control systems in the cloud, based on quality metrics, formulated by NIST. Moreover, we discuss best strategies and existing mechanisms to enforce policies based access control in a cloud.

3.1 Secure Data Storage in Cloud

Securing data in a cloud storage becomes critical requirement, while amount of sensitive data stored in a cloud increases. Number of theoretical and practical solutions exist in the literature to address this problem[63, 22, 28, 25, 17, 66]. Below we will describe three existing works of our choice, to present examples of three, conceptually different approaches.

Kumbhare et al.[29] proposes a secure cloud storage repository - Cryptonite, with low key management overhead and client-controlled security. Cryptonite service utilizes a concept of StrongBox file, to achieve better security and performance over plaintext storage. StrongBox file enables scalable key management by securing multiple files that share the same permissions using just the single global public-private key pair for each user. It relies on broadcast encryption and uses several standard cryptographic techniques in its design. Cryptonite service leverages the Public Key Infrastructure (PKI) to provide authenticity, integrity of a stored data, and non-repudiable auditing. Cryptonite repository service provides REST service interface and asynchronous data storage capabilities.

In their research, Gupta et al.[17] address the problem of trust between data owner and cloud service provider(CSP), by eliminating trust requirement between them. They propose a secure data storage scheme where security of the data will be controlled by data owner only. Data owner specifies access rights for his/her own data and manages revocation, if any. Users may search the files in an encrypted database, in a secure manner, with the help of ranked keyword search. Their architecture follows a Client/Server model, where client performs all cryptographic operations, whereas server performs search operations over the encrypted data. Additionally, client application notifies the data owner in case of any security breaches.

Popa et al [41] propose secure storage system - CloudProof, specifically designed for the cloud. The main goal of this system is to notify cloud users about integrity violations, regarding their data in cloud, and to provide proves of violation occurrences to a third party. They believe in efficiency of proof-based system in enabling security guarantees in Service

Level Agreements(SLA), where customers pay for a certain level of security and suppose to receive compensation in case of security incidents originated from cloud provider. Additionally, CloudProof provides scalable access control to the data, while maintaining the performance and availability of the cloud services. Broadcast encryption is used as a cryptographic technique to encrypt the data for a group of the legitimate users, similar to the work from Kumbhare et al.[29].

Most of the existing work in this area is based on advanced cryptographic techniques and developed as a proprietary solutions. The lack of standardized interfaces results in interoperability and scalability issues, making these solutions hard to adopt under various use cases.

Moreover, cryptographic operations for big files may be time consuming on the clients local machines, whereas performing necessary cryptographic operations in the cloud may decrease the time needed for computations [17] (see 2.1). Performance of the system and security of the data should be balanced, depending on the use case.

3.2 Access Control Systems in Cloud

Access control requirements in the cloud differ from case to case, since cloud environment has to offer services to various customers. Number of access control models and systems has been developed for the cloud to address certain requirement sets [60, 58, 62, 55, 49, 5].

Role Based Access Control (RBAC)

A. Sirisha and G. Kumarihave have proposed RBAC for the cloud [51]. According to their work access control is performed in two steps. First user is authenticated based on provided attributes. Then user roles are identified and corresponding access rights are assigned to the user. If user is already registered in the database, authentication is performed by validating attributes via identifier. Otherwise, authentication may be achieved by validating credentials provided by the user. Similarly, permissions are assigned to the roles based on identifiers.

Task-Role Based Access Control (TRBAC)

TRBAC proposed by H. Andal and M. Hadi [36]. TRBAC separates roles from the tasks, opposed to RBAC, where roles and tasks are combined. Tasks are classified based on active and passive access control. Additionally, tasks may be inheritable or non-inheritable. Active access control is required for tasks that are part of the workflow. Conversely, tasks, that are not a part of workflow require passive access control. In TRBAC users are assigned to roles, roles are assigned to tasks, and tasks are assigned to permissions.

Attribute Based Encryption Fine-Grained Access Control(ABE FGAC)

Li et al. proposed a fine grained access control in cloud, based on attribute based encryption in their work [31]. This access control mechanism provides better accountability and user revocation. Two types of ABE exist according to this model. The first is key policy ABE (KP ABE), where access policy embedded to the private key, which is assigned to the users. This key can decrypt the files that match the policy attributes with embedded policy. The second is ciphertext policy based ABE (CP ABE), where access policy embedded to the cipher text

with each file. User key has different attributes, defining the structure of the access. The user may access the file if attributes match the structure of the file. The system consists of data owners, users, cloud providers and third party auditors. Broadcast encryption is used by data owner to define user groups that may access his/her files. User may access the file, if number of attribute values is matched with the policy, is at least equals to predefined threshold value.

Fine-Grained Data Access Control(FGAC)

Wang et al. propose a hybrid access control model in [64] utilizing attribute based encryption (ABE), proxy re-encryption and lazy encryption. Files contain attributes and public key corresponding to these attributes. Access structure of files is defined by the logical expressions over public key attributes. Data file sets are defined for each user. Files are encrypted by symmetric keys. Symmetric keys are encrypted with attribute based encryption corresponding to key policy. To revoke a user, data owner specifies minimal set of attributes and modifies public and master key according to the specified attributes. Then he generates proxy re-encryption keys. Data owner sends user ID, attribute set, proxy re-encryption keys and public keys to the cloud server. Cloud server revokes the specified user, and stores updated keys in attribute list.

Hierarchical Attribute Based Encryption Access Control (HABE)

Hierarchical attribute based encryption has been proposed by Wang et al.[61] as an access control system, by combining hierarchical identity based encryption (HIBE) and ciphertext policy based attribute based encryption (CP ABE).Hierarchy consists of a root master (RM) and domain masters (DM). RM is responsible for generation and distribution of keys.DM is responsible for providing necessary attributes, and handling delegation and distribution of the keys to the users. Each user has ID and certain attributes. User's position is determined by ID and public key of the DM administrating the user.

National institute of standards and technology (NIST) has formulated the quality metrics for the access control systems [20]. Um-e-Ghazia and Masood [12] have performed evaluation of aforementioned access control systems in the cloud, in accordance with quality metrics proposed by NIST. The results of this analysis is shown in Table 3.1

Characteristics	RBAC	TRBAC	ABE FGAC	FGAC	HABE
Least Privilege	medium	high	high	high	high
Separation of duty	medium	high	high	high	high
Complexity	low	low	high	high	high
Performance of Enforcement Mechanism	medium	medium	low	low	low
Policy Conflicts	high	high	high	high	high
Horizontal Scope	high	low	low	low	medium
Configuration Flexibility	high	low	low	low	low

Table 3.1: Comparison of access control systems on cloud [12].

Their analysis shows that all analyzed access control systems aimed to support different aspects of cloud authorization, and provide solution to any one problem of access control in the cloud. None of these access control systems covers complete aspects and requirements of cloud platform.

3.3 Policy Based Access Control in Cloud

Hu et al. presented a new semantic access control policy language (SACPL) to describe access control policies in cloud computing environments[21]. They propose an ontology-based access control, to support interoperability between various distributed access control policies. Their research includes analysis of interoperability between different access control schemes and models.

XACML is a common, accepted standard for describing access control policies. However, XACML does not specify the transmission mechanisms of the authorization messages. Therefore, we did a literature review on existing implementation strategies of XACML.

In their work, Lorch et al. [34] presented XACML as a component of a distributed authorization framework. They discuss several distributed systems incorporating XACML, and present their high level architectural view. Discussions of deployment scenarios, for XACML-based authorization services take place in their paper. Their research presents an example of how XACML may be implemented in a production environment.

XACML may provide functionality of Access Control Lists (ACL) and maybe integrated to the existing access control systems, to extend access control features of these systems, while keeping their original settings and backward compatibility.

Karjoth et al. map XACML to ACLs in their research, conducted at IBM Research in Zurich Research Laboratory [27]. This was done to safely add policy based access control to widely used legacy access control system, utilizing ACLs as a major access control mechanism.

3.3.1 Data Centric Policies

As was described in Section 2.4, decentralized and sovereign access control is needed in the cloud, to support the diverse object types and heterogeneous environment. Moreover, data usage may be a concern, once it leaves the boundaries of the cloud and resides in the local system of consumers. Data-centric policies may provide flexibility to address these challenges. Typically, data-centric policies can be attached to the data and enforced on time of use. Data-centric security model is depicted in the Figure 3.1.

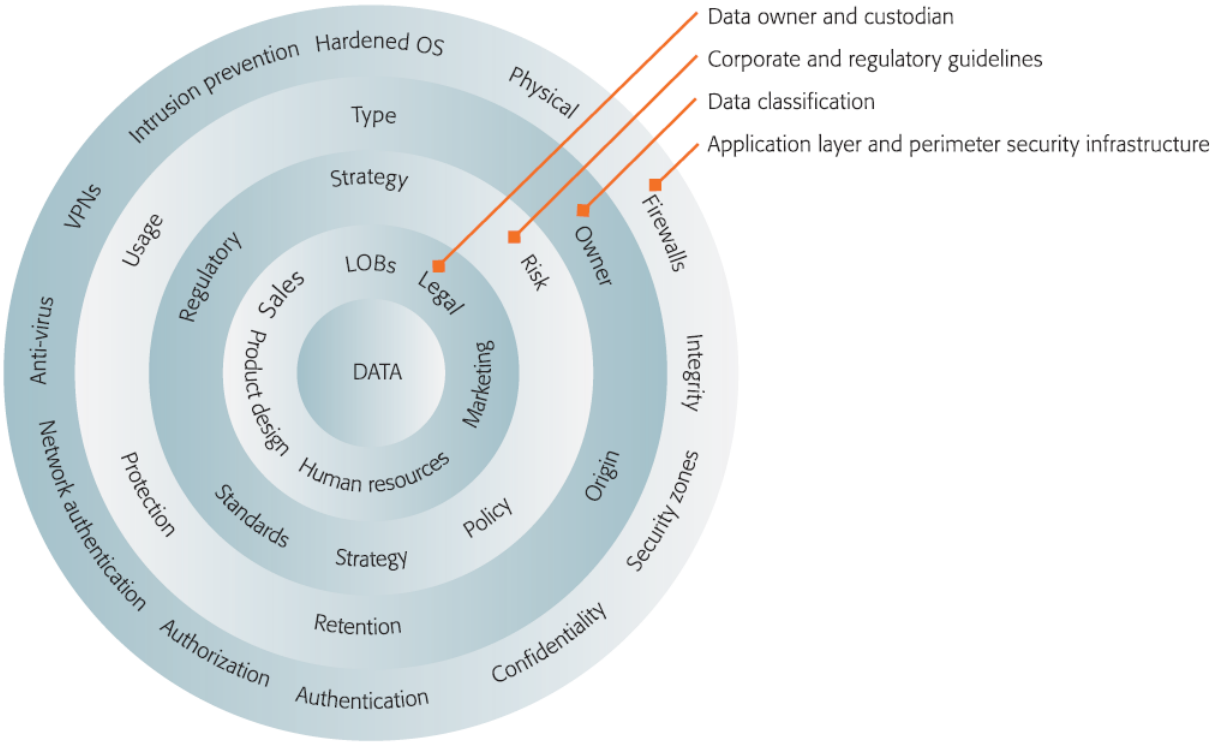

Figure 3.1: Data-centric security model.(Figure taken from [19])

In their paper [54], Speiser and Harth describe the advantages of data-centric policies, on the example of Smart Grids. They propose a data-centric view to this system, where policies are attached to the data, and restrict isolated uses of data to the data directly. The main benefit of this solution is applicability in scenarios without centralized control. In their work, Speiser and Harth proposed attachment of data-centric policies to the information artefacts in the form of sticky policies. Sticky policies may be attached in two ways: attached by value and attached by reference. By value means, that policies are stored and transported together with an artefact. By reference means, that policies may be retrieved based on identifiers.Particularly, policy identifier is attached to the artefacts. At the time of usage, policy is obtained based on attached identifier, and usage request is evaluated agains this policy.

Furthermore, Trabelsi and Sendor implemented sticky policies in a cloud environment, and described results in their paper [57]. Policies are attached by value to the data. However authorization access is managed via intermediate security service, called SPACE. SPACE is based on the sticky policy technology and offers access and usage control functionalities to

the data anywhere in the cloud.

Chapter 4

Preliminaries

In this chapter we introduce relevant preliminary knowledge that is important in understanding of system architecture and its core concepts. We briefly explain architectural style, standards and specifications used in current work, with emphasis on the most utilized features.

4.1 Rest Architecture

Representational State Transfer (REST) is an architectural style, mostly used for web services and large-scale distributed systems. It specifies certain constraints, that can result in better performance, scalability, simplicity and other desirable properties if implemented properly.

All data and functionality are considered as resources in current architectural style. These resources can be accessed by Uniform Resource Identifiers (URIs). Resources are affected or triggered by simple and predefined operations.

The REST architectural style adopts client/server architecture. It is stateless by the nature, and typically operates over HTTP.

The REST architectural style is based on following principles, that are providing simplicity, performance and scalability to the RESTful applications:

- **Resource identification through URI:** A RESTful web service provides certain available resources to the clients and all interactions are performed using these resources. Typically, URIs are used to identify the resources, since they may provide globally unique identification of resources and services.
- **Uniform Interface:** Four operations are used to manipulate resources in REST architecture. In particular, these operations are create, read, update and delete (CRUD) operations. Resources may be created by using PUT operations. GET retrieves the state of a resource in standard representation. DELETE operation is used to delete a resource. Finally, POST transfers a new state onto a resource. POST might be used to create resources, if, command is sent to the server to create a subordinates of these resources, based on server-side algorithms. Typically, architecture promotes the simplified CRUD to REST mapping.
- **Self-descriptive messages:** Content of resources may be accessed in a various formats, since their representations may differ, based on requests parameters, typically headers.

Some of the well known formats are HTML, XML, PDF, JPEG, JSON, Base64, plain text and others. Metadata of the resource can be used to provide additional information about a resource, its representation, access control, etc.

- **Stateful interactions through hyperlinks:** REST architecture supports only stateless resource interaction, meaning that every message is self-contained. In other words, every message carries all needed information explicitly, and server is not supposed to keep any information about the communication, including its state. Techniques like URI rewriting, cookies and hidden form fields can be used to exchange state between nodes.

Furthermore, REST architecture supports caching, clustering and load balancing as a built in features. These features allow web services easily scale up and serve a very large number of clients.

4.2 Attribute Based Access Control

Attribute-Based Access Control (ABAC) [30] is an access control model where access rights are granted to the users based on attributes. These attributes are associated with the requested resource or the requester, and combined in the policies. ABAC provides mechanisms for each information owner to specify their own policies and combine policies resulting in final authorization decision. ABAC can be considered as a generalization of Role-Based Access Control (RBAC).

While requesting access to a resource, client provides a set of attributes, which will be checked against attributes required to grant access, by the access control system. If provided attributes meet the requirements to perform requested action, access to the resource will be granted.

The main advantage of the ABAC is its suitability for a distributed environment. Client requesting access does not need to be registered in the system beforehand to be able access certain system resources. Hence, access control is decentralized, and information owners may enforce fine-grained control of their own resources. This feature is significantly reduces administrative burden for access control in the system where users may leave or join the system arbitrarily.

Metadata is a critical consideration while implementing ABAC system. All services and information should be classified appropriately, in order to be accessible based on provided attribute sets. Classification may involve tagging or marking with appropriate metadata.

ABAC is not supported by most of the operating systems and commonly implemented at the application level, by adopting widely accepted eXtensible Access Control Markup Language (XACML) standard.

4.3 Cloud Data Management Interface

The Cloud Data Management Interface (CDMI) is used to create, retrieve, update, and delete objects in a cloud. The features of the CDMI include functions that

- allow clients to discover the capabilities available in the cloud storage offering;

- manage containers and the data that is placed in them; and
- allow metadata to be associated with containers and the objects they contain.

This international standard supports two types of operation. Particularly, operations that use CDMI content type in the HTTP body and operations that use standard content type in the HTTP body. Support of both types makes it possible for both, CDMI-aware and non-CDMI-aware clients to interact with CDMI provider. CDMI provides mechanisms to manage containers, domains, security access , and monitoring/billing information. These mechanisms are supported even for proprietary protocols, by exposing underlying data services to the clients.

Since CDMI is a broad standard, most of the cloud service providers may support only a subset of it. This can be achieved by exposing limitations of the cloud provider via supported capabilities.

CDMI uses principles of RESTful architecture in the interface design. It clearly defines data management, storage and retrieval procedures. For data operations, the client should know about container objects and data objects. Containers act like folders in a typical storage systems, while providing abstraction from the underlying storage protocols, to the clients. iSCSI, CIFS, NFS are few examples of supported storage protocols. HTTP PUT, POST, GET and DELETE requests are used to manage data according to REST principles.

CDMI supports queue objects, that are playing critical role while concurrent data operations, by providing in-order, first in, first-out creation and fetching of queue values.

CDMI Metadata

CDMI uses various types of metadata, including HTTP metadata, data system metadata, user metadata, and storage system metadata. Metadata provides useful information for the CDMI clients, such as data encoding, data size, data type, etc. Metadata may be used to increase data search speed in a big data stores, and capability to search through the encrypted data, by containing extra information. CDMI data system metadata, user metadata, and storage system metadata is defined in the form of name-value pairs.

CDMI specification mandates usage of SSL/TLS to secure communication between CDMI entities and strongly encourages the usage of most current versions of these protocols.

4.4 Key Management Interoperability Protocol

The Key Management Interoperability Protocol(KMIP) [43] is an OASIS standard, which specifies the communication protocol between clients and a key management server to perform various key management operations on the cryptographic objects managed by the server.

By defining a low-level protocol, KMIP enables single, interoperable key management infrastructure, where any KMIP-aware client may perform key management operations with any KMIP-aware key manager.

KMIP specifies three element types:

- **Objects** are the cryptographic objects upon which operations are performed. The KMIP specification includes two types of objects, base objects and managed objects. Base objects deal with interactions between client and server. Managed objects are the

cryptographic objects, predefined templates and metadata. The cryptographic objects include symmetric keys, asymmetric keys, digital certificates, secret data, split keys and opaque objects.

- **Operations** are the actions performed with regard to the objects, such as registering an object in a key management system, modifying attributes of an object, etc. KMIP specifies certain operations handling life cycle of the cryptographic objects, such as Create, Register, Destroy, Archive, Revoke, Activate, Deactivate and Recover. Additionally, KMIP specifies operations dealing with searching and retrieving cryptographic objects, including their attributes.

- **Attributes** are the properties of the cryptographic object, such as the type of the object, its unique identifier, usage, owner, etc. Each cryptographic object is coupled with a subset of supported attributes. Additionally, KMIP defines the structure, format and legal values for the attributes. Furthermore, KMIP might enforce access control over the attributes , by defining who can read, modify or delete certain attributes.

KMIP defines the states representing the life cycle of the cryptographic object, by using attributes related to the key state, based on NIST special publication 800-57 [13]. Each cryptographic object should be in one of the states at any given time. Particularly, these states are Pre-Active, Active, Deactivated, Compromised, Destroyed, Destroyed/Compromised.

KMIP specification mandates usage of SSL/TLS to negotiate a secure connection between clients and key management system.

4.5 eXtensible Access Control Markup Language

eXtensible Access Control Markup Language (XACML) is an XML-based, general purpose policy system for access control that has been standardized in OASIS.

XACML defines access control policy syntax, request/response syntax. Subsequently, it defines semantics for processing access control policies and determining applicability of policies to requests. Particularly, XACML defines language primitives to describe rules, policies, functions and data types. XACML may support Access Control Lists (ACL), fine grained policies for distributed systems, etc. , by using these primitives. Since XACML provides a common policy format, it can be used to share or exchange access control policies between multiple security domains.

XACML defines the structure of request and response messages and provides a standard interface between Policy Enforcement Point (PEP) and Policy Decision Point (PDP). A typical request consists of attributes representing requesting subject, requested resource, intended action and the environment. A response may contain following four decisions: Permit, Deny, Not Applicable, Indeterminate. XACML supports provision of obligations via response messages. Obligations are the directives from the PDP to the PEP, originated from applicable policies. PEP may grant access only in compliance with that directives.

XACML defines data types, such as string, boolean, integer, time, set, etc. Furthermore, XACML defines functions to perform certain operations on these data types, such as arithmetic functions, comparison functions, etc.

Policies may include references to other policies, resulting in support of distributed, decentralized rules, where each rule can be managed by separate organization. XACML uses

number of standard, predefined policy combining algorithms to combine multiple rules or policies while evaluating authorization requests. Moreover, XACML defines a standard extension mechanism to develop new policy combining algorithms.

Chapter 5

System Architecture

In this chapter we introduce overall design and architecture of the proposed system, enforcing policy based access control to the data stored in the cloud. We discuss various aspects of the system and essential requirements for its proper functioning. Furthermore, we explain conceptual decisions made while designing the system. Finally, we summarize the operations flow and corresponding sequence diagrams.

5.1 Architecture

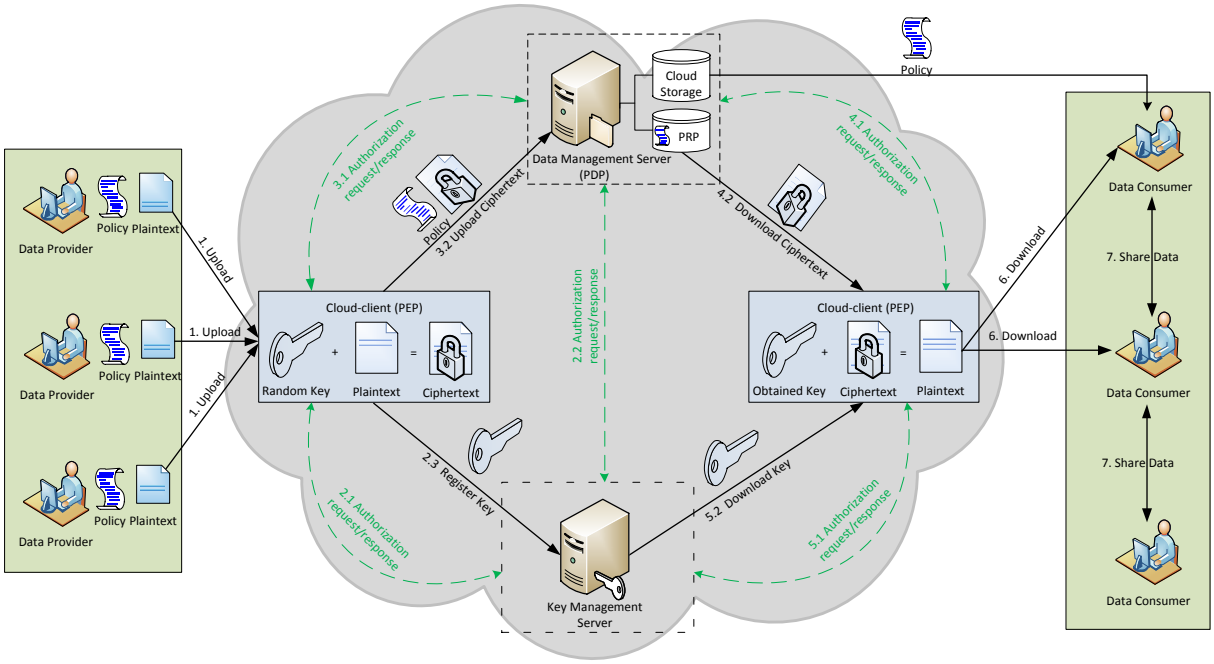

Figure 5.1: Architecture of the cloud data sharing system.

This section presents the architecture for policy enforcement in the cloud, which combines secure data storage, key management and policy management features together. Figure 5.1 presents the high level architecture of the proposed system, supporting secure data

sharing in a cloud environment. The main goal of the system is to support data sharing over a cloud environment while keeping data confidentiality and integrity and limiting access and use to authorized parties only. The architecture consists of several components that handle the process flow of the system.

5.1.1 Users

There are two user roles in the current system architecture:

- **Data providers** are user roles having direct access to the cloud data sharing system. They provide data that users may obtain from the cloud data sharing system and process.
- **Data consumers** are user roles obtaining and processing data from the cloud data sharing system. They have direct or indirect access to the stored data. Indirect access to the data is possible by sharing the data between data consumers.

Users may act on behalf of both roles, depending on the actions they perform. Users may specify and upload data-centric policies related to the data which they are intended to upload, while acting as a data provider. Users should act according to the data-centric policies related to the data, while processing data outside the cloud, and acting as a data consumer. This behavior is enforced by establishing trust relationship between users of the data sharing environment [42].

5.1.2 Cloud-client application

Users will interact with the cloud data sharing system through a Cloud-client application, which is also deployed in a cloud environment. Literally, Cloud-client application is a front-end service that performs most of the critical operations in the given architecture, as described below.

Most of the data operations (see Section 5.6) are performed only through Cloud-client application, such as data upload and data download. However, certain operations are done without its intervention, such as data operations outside cloud. Operations, such as key generation, registration and retrieval are also performed only through the Cloud-client application.

Cloud-client application is responsible for authenticating users before the start of any operations involving its intervention. Handling authentication process by the application deployed in the cloud is efficient, due to the high available and scalable nature of the cloud computing.

Moreover, the Cloud-client application performs cryptographic operations while data upload and download. Cryptographic operations can be done more efficient in a cloud environment, with a flexible resource allocation (see Section 2.1) and extensive computing power. Additionally, this approach simplifies user interaction with the system by offloading interoperability considerations from the user side.

Furthermore, Cloud-client application performs an authorization requests on behalf of the authenticated users and enforces the decisions made by a Policy Decision Point (PDP). Particularly, authorization requests to the PDP are done while data upload/download and key registration/retrieval. In other words, it acts like a Policy Enforcement Point (PEP) in a given architecture, as can be seen from Figure 5.1.

5.1.3 Data Management Server

All data is stored in the cloud storage and managed by the Data Management Server (DMS), which is deployed in a cloud environment. DMS handles data upload and download requests performed from the Cloud-client application, and provides mechanisms to support other data operations requested outside the cloud, such as sharing of the data with certain third parties, by providing sticky policies (Section 3.3.1). Additionally, it performs integrity and authenticity checks while data upload operations.

Data stored in the cloud storage includes all types of policies supported by the given system. Policies are stored in the Policy Retrieval Point (PRP), which is also a part of the cloud storage, but explicitly depicted in the Figure 5.1 to present a complete set of the components of access control used in this system, to the reader.

Additionally, DMS performs tasks of PDP in the current system architecture. The main reason for this design choice is that all policies are stored in a cloud storage managed by this component, which simplifies their retrieval and reduces communication overhead in the network. Another reason is to minimize the components involved in the architecture and reduce the complexity of the system. Lastly, this design decision simplifies implementation, testing and maintenance of this component.

As an alternative, PDP might be deployed in combination with PEP at the Cloud-client application. However, we do not support this design choice, for the following reasons. In a typical scenario, PEP is assumed to intercept requests at the edge of the system, and deployed at the closest location relatively to the source of the request. Cloud computing provides abstraction of the underlying services and high availability to the users. Subsequently it may seem that PEP is deployed in a single location from the user point of view. However, in reality cloud is composed from multiple, interconnected data centers across the world. Obviously, it is more effective to store policies centrally to avoid the burden of synchronization between dislocated storages. Thus, aforementioned assumption holds for a a cloud environment too.

Moreover by deploying PEP and PDP separately, we provide an externalized approach to authorization. This provides a flexibility for an individual information systems, that may require to query the PDP via their own PEPs.

5.1.4 Key Management Server

All keys are stored and managed by the Key Management Server(KMS), which is deployed in a cloud environment too. KMS handles key registration and retrieval requests performed from the Cloud-client application only, since KMS does not provides authentication for consumers, and relies on trustworthiness of the authentication service on the Cloud-client application. In case of key registration, KMS responds with randomly generated Unique Identifier for the given key, to the Cloud-client application.

All stored keys may be encrypted by the KMSs master key. Subsequently, KMS may decrypt the requested key and sends it to the requester, while key retrieval process. Additionally, KMS sends Pair Verification Message(PVM) to the DMS while key upload process. PVM is simple a message to check if the key corresponds to proper Unique Identifier of the key, and will succesfully decrypt the data at the time of use. It contains hash of the concatenated values of Unique Identifier of the key and the key itself. This is done to provide DMS with sufficient information to perform integrity and authenticity check over the subsequently up-

loaded data. PVM assures that the data was encrypted with the same key, registered under a given Unique Identifier in KMS. Details of this process are described in the Section 5.6.1.

Additionally KMS forwards authorization requests to the PDP and forwards decisions to the requester, while key registration and retrieval operations. In other words, it acts like an authorization proxy. This design decision increases communication overhead in simple scenarios, since direct authorization request to the PDP is more efficient. However, it provides more flexible and customizable solution in scenarios when KMS should add additional information to the authorization request, such as attributes characterizing the circumstances (e.g. local time, state of the KMS, etc.).

5.2 Authentication

Authentication of the users is done by the Cloud-client application, as was mentioned in Section 5.1.2. Various authentication mechanisms [6, 26, 44, 11] might be applied in current system architecture, and system does not biased towards any of the existing solutions. Authentication may be performed either at the message level or at the session level. Proper security analysis should be done before choosing applicable authentication mechanisms, considering the use case scenario.

However, there are essential requirements for an authentication process in current architecture, to support its main goal (data confidentiality and integrity).

- Users should authenticate Cloud-client application while data upload operation. Otherwise users may upload plain confidential data to the untrusted location.
- Cloud-client application should authenticate users while data download operation. If this requirement is unfulfilled plain confidential data may be issued to the untrusted parties.
- Cloud-client application should authenticate KMS, while key registration process. Otherwise decryption key may be compromised, which can leak confidential data.
- KMS should authenticate Cloud-client application, while key retrieval process. Otherwise decryption key may be issued to the untrusted parties, which may lead to the leakage of confidential data, depending on how keys are managed (Section 5.4).

It is worth to note that, aforementioned requirements for an authentication process are not enough to maintain proper functionality in a real system. Requirements as authenticity and availability of the data should be taken into account too. Therefore, mutual authentication is recommended between all components of the system. Mutual authentication may provide better usability and significant prevention of DoS attacks in a real systems, by preventing storage of extensive bogus data and its subsequent retrieval. However, it is not sufficient without improper design of other components of the system, such as authorization.

5.3 Authorization

Authorization in the current architecture is performed based on predefined or user-defined authorization policies. In other words, Policy-Based Access Control (PBAC) is enforced in

current system architecture. PBAC is a model that standardizes the ABAC model, by extending it with complex extensions, and supporting a wide range of policy models, including RBAC as a specialization of ABAC. PBAC supports more fine-grained policies that control access to the data based on provided attributes and circumstances. More often, enterprise level organizations implement PBAC over ABAC to achieve tighter control and auditability over resources, where they should also comply with certain legislations.

Authorization policies in the system are stored either, in the Policy Retrieval Point (PRP), or in the local IT systems of the users. These policies can be classified in a following way:

- **Server-side policies:** Authorization requests performed by the Cloud-client application are evaluated against these policies. Server-side policies are administered by the Policy Administration Point (PAP). Location of the PAP is not predefined in the current system architecture and can differ between use cases.

- **Data-centric policies:** Authorization requests performed from sources other than Cloud-client application are evaluated against these policies. An authorization request to share the data may be an example for this case. Data-centric policies are attached by the reference in current architecture, and may be determined by the data provider while uploading the data.

In order to provide scalable policy management, system utilizes the concept of PolicyMapping file. This file consists of references to the data-centric policies, and makes possible for multiple data files to share the same policy file. PolicyMapping file simplifies policy changes for the particular class of the data, since changes may be applied to the group of data files, rather than individual data files. Additionally, PolicyMapping file reduces the storage space requirements by eliminating policy duplications.

5.4 Key management

Key operations are performed only by Cloud-client application and KMS in the proposed system architecture. Only keys generated by Cloud-client application are stored in KMS.

All stored keys are encrypted by KMS's symmetric server key. KMS's server key never leaves KMS and it is kept secret, otherwise all data encryption keys may be compromised.

Cloud-client application does not keep generated encryption key after its successful registration in the KMS, and successful data encryption process. Furthermore, Cloud-client application does not keep decryption keys after successful decryption process, in order to decrease the risk of key compromise.

Key transmission operations, such as key registration and key retrieval, are performed over secured channels only, according to the number of standards and specifications(e.g. KMIP specification [43]).

5.5 Encryption

All data and keys are stored in an encrypted form in the current architecture. Only symmetric keys generated by Cloud-client application are used to encrypt the data. Keys are encrypted and stored in the KMS, using symmetric server key of the KMS.

All data and keys are transmitted over secured channels. There are essential requirements for the secure channels in current architecture:

- Channel should provide data confidentiality, so that the secrecy of the data is guaranteed.
- Channel should provide data integrity, so that data is not forged on transit.
- Channel should be replay resistant, to prevent data replication and subsequent negative effect.

Depending on the requirements of the use case and size of the processed data, stream or block ciphers can be used while data encryption/decryption process.

5.6 Data Operations

This section describes overall data operations in the given architecture, and shows how all components of the given architecture are interoperates, summarizing overall process flow. Note that only successful flow of data operations is described in this section. Handling of unsuccessful flow of data operations might differ between use cases. It worth to note that all communications in the system are synchronous.

5.6.1 Data Upload

Diagram 5.2 describes the steps of the data upload process in the current architecture. Authentication between components is not included into this diagram, to not distract the reader from main process flow and to reduce the size of the diagram. All interactions between components are possible only after sufficient authentication process, as described in the Section 5.2, and over the secured channels, as described in the Section 5.5.

For a data provider to upload a plaintext data file PD to the cloud storage, it first performs mutual authentication process with the Cloud-client application (PEP). If authentication is successful, Cloud-client application accepts plaintext data file and associated policy file from the data provider. Then, it generates a random symmetric key SymKey, and requests the KMS to verify data provider's authorization, based on supplied attributes, to perform given operation (upload key to KMS). Authorization request at least contains data provider's credentials and action to be performed. Since, there is a single PDP in our architecture, KMS forwards this authorization request to the DMS (PDP), which evaluates the authorization request, and responds to the requester (KMS). KMS forwards authorization decision to the Cloud-client application.

If decision is positive, Cloud-client application requests KMS to register the SymKey. In case of success KMS generates Unique Identifier for the provided SymKey - keyID, and saves the SymKey in the file with the name of generated keyID, by previously encrypting it with KMS's secret key.

Later, KMS calculates PVM (Section 5.1.4), by taking hash of the SymKey in combination with keyID, and sends it to the DMS, along with the keyID. If operation is successful, KMS sends keyID to the Cloud-client application through the previously established secure connection.

Cloud-client application requests the DMS to verify data provider's authorization, based on supplied attributes, to perform given operation (upload data to the cloud storage). Authorization request at least contains data provider's credentials and action to be performed. In case of successful authorization, Cloud-client application calculates PVM based on SymKey and keyID it holds. Then, Cloud-client application adds associated policy file's name, keyID value and PVM value to the metadata of the data file. Finally, Cloud-client application encrypts data with SymKey and uploads data to the cloud storage along with its metadata and associated data-centric policy, if specified. Note that, upload sequence is not important, therefore, policy file is uploaded first due to the specifics of implementation.

DMS verifies keyID and PVM values, by comparing them with the values, previously uploaded by KMS. If verification is successful (i.e. key-keyID pair is verified), Data Management Server stores the data, its metadata and the associated policy file. Additionally, DMS adds the reference for the stored policy file to the "PolicyMapping" file (See Section 5.3), if needed. If PVM check is unsuccesful, DMS deletes data, its metadata and associated policy file.

5.6.2 Data Download

Diagram 5.3 describes the steps of the data download process in the current architecture. Authentication between components is not included into this diagram, to not distract the reader from main process flow and to reduce the size of the diagram. All interactions between components are possible only after sufficient authentication process, which is described in the Section 5.2, and over the secured channels.

For a data consumer to download a data, it first performs a mutual authentication process with the Cloud-client application. If authentication is successful, Cloud-client application requests the DMS to verify data consumer's authorization, based on supplied attributes, to perform given operation (download data to the cloud storage). Authorization request at least contains data consumer's credentials, data identifier, and action to be performed.

If decision is positive, Cloud-client application requests DMS to download the data. DMS responds with encrypted data and its metadata. Metadata contains keyID and reference to the data-centric policy of the file (Diagram 5.4), if applicable.

Then, Cloud-client application requests the KMS to verify data consumer's authorization, based on supplied attributes, to perform given operation (download key from KMS). Authorization request at least contains data consumers credentials and action to be performed. KMS forwards this authorization request to the DMS (PDP), which evaluates the authorization request, and responds to the requester (KMS). KMS forwards authorization decision to the Cloud-client application.

If decision is positive, Cloud-client application requests KMS to obtain a decryption key SymKey, based on keyID contained in the metadata of the data. Since all keys stored in the KMS are encrypted, KMS decrypts the encrypted SymKey before responding it to the Cloud-client application.

Finally, Cloud-client application decrypts the data, using SymKey and provides it to the data consumer, along with its metadata.

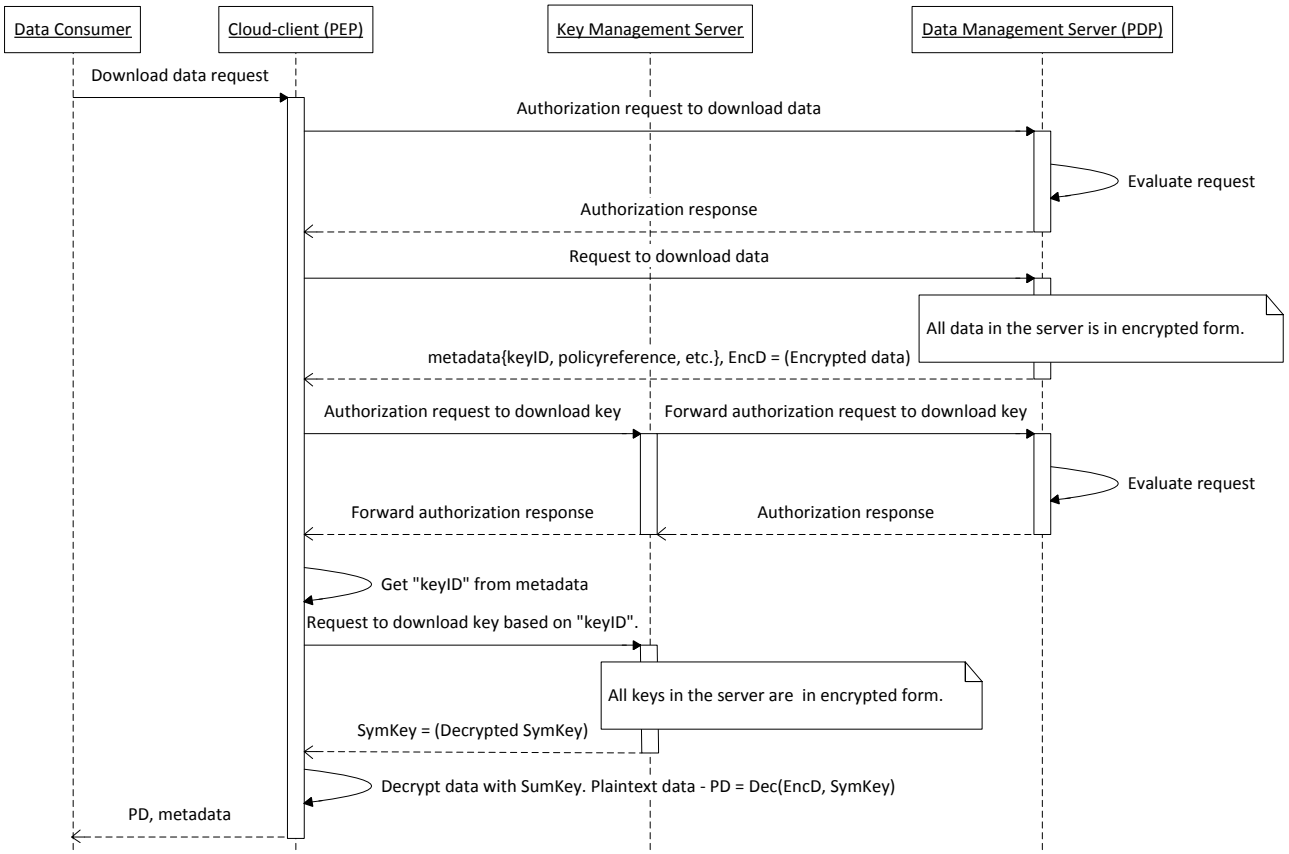

Figure 5.3: Data download process.

5.6.3 Data Operations Outside Cloud

There are number of use cases, that might require distinct data operations once data is released to the consumer. As was mentioned before, these operations should comply with specified data-centric policies. As an example, we will describe the data sharing operation outside the given cloud data sharing system. Diagram 5.4 describes the steps of the data sharing process in the current architecture. Consumer in this case indicates local IT system of a system user, performing data sharing.

For a consumer to share a data, it first obtains reference to the data-centric policy of the data, from the metadata. Then, consumer requests DMS to download data-centric policy file from DMS, based on the obtained policy reference.

After obtaining policy file, consumer evaluates data sharing request against policies prescribed in this file. If evaluation results in positive outcome, consumer performs sharing operation.

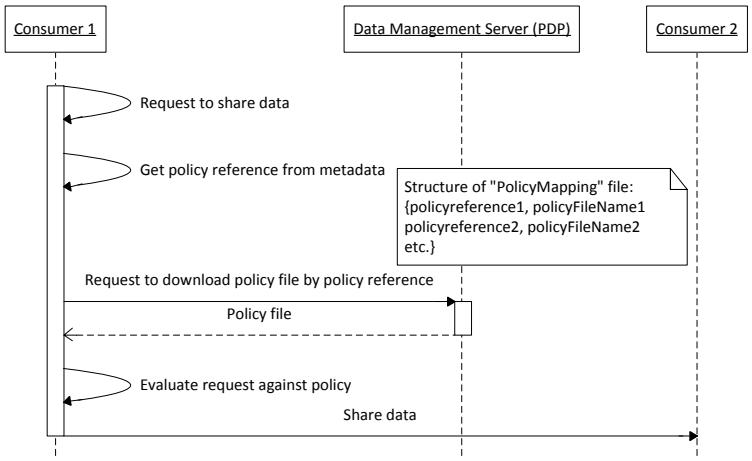

Figure 5.4: Data sharing process.

5.6.4 Additional Notes

Functionality of the system can be extended unforeseeably, by adapting to the various use cases. The main objective of the current system design is to provide solid building block, describing policy enforcement in and outside cloud, for the systems to be developed on base of current architecture.

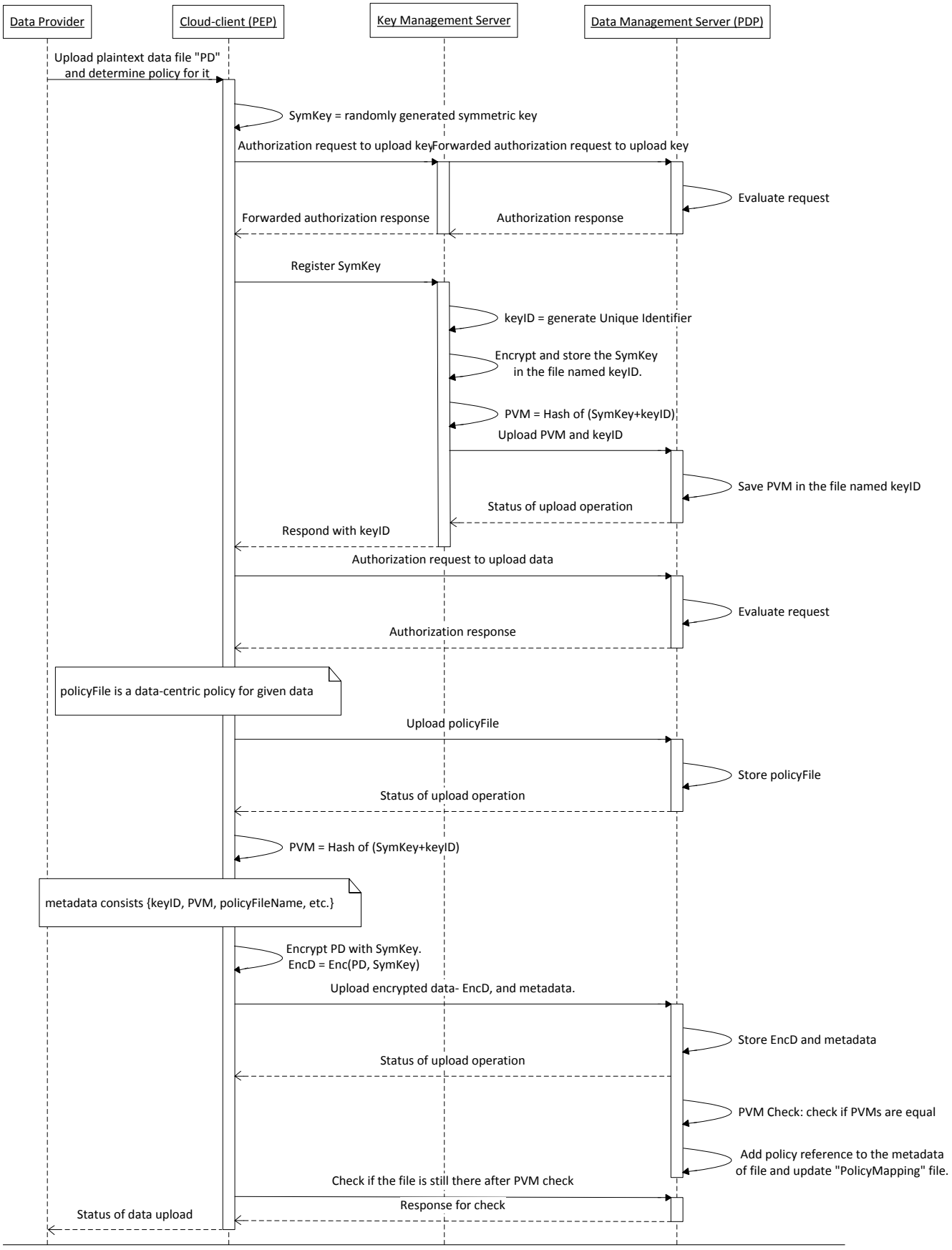

Figure 5.2: Data upload process.

Chapter 6

Implementation

In this chapter we introduce the implementation of the proposed architecture. We specify the involved components, and their roles in the current architecture. Furthermore, we explain standards and technologies used in this implementation. Some of the most noticeable challenges and implementation issues are also mentioned in this chapter.

6.1 Implementation Structure

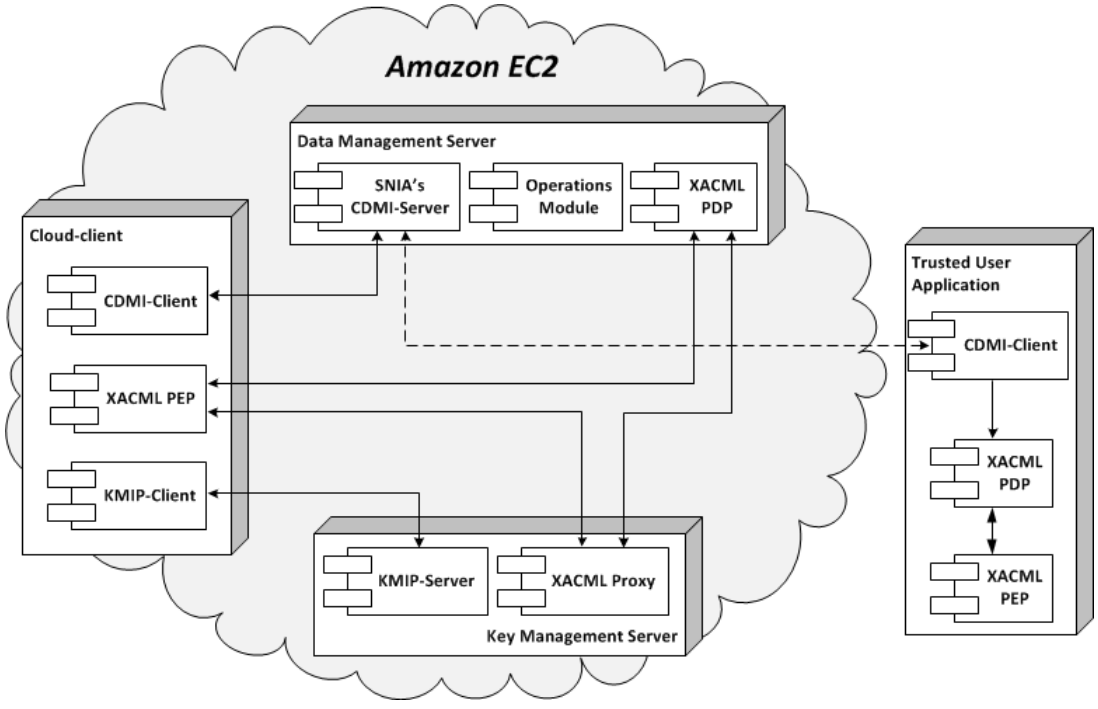

Figure 6.1: Implementation architecture of the cloud data sharing system.

The architecture of the cloud data sharing system consists of several components. Each component consists of multiple modules. In this implementation a module is an application, performing certain specific tasks. To show the feasibility of the proposed system, we

have created an implementation of the architecture. Figure 6.1 shows the structure of the implementation.

All components of the system are developed in the Java programming language [16]. Java is a cross-plaftorm programming language, meaning that compiled code runs on multiple platforms without a need of recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture.

Currently, our implementation supports data upload, download and sharing functionality. This functionality includes proper key management and policy enforcement mechanisms. Functionality of our implementation can be extended and tailored according to the needs of specific use cases.

We chose Amazon Elastic Compute Cloud (Amazon EC2) [4] as a cloud computing environment for our implementation. Amazon EC2 is a web service, that provides resizable computing resources in the cloud. It is managed through the native web interface, and lets you to scale the resource capacity up or down in a short time period. It is easy to use, and reduces the time to install and run a new servers. Application and web hosting are the typical use cases of Amazon EC2.

All system components, except Trusted User Application (TUA) are deployed in the Amazon EC2 and involved to the data upload/download process. TUA is deployed locally, on users' site and provides support for data operations outside cloud, particularly data sharing.

Following sections will describe implementation details and specifics of each component and its modules, divided into functionality groups.

6.2 Authentication

Implementation of authentication mechanisms is out of scope of this thesis. Basically, various authentication mechanisms can be integrated to the system. Preferably, authentication should be achieved on the Cloud Client, since it is the first point of interaction with the users of the system.

6.3 Policy Enforcement

Policy enforcement is done by three components in the current architecture. Particularly, these components are DMS, Cloud-client application and TUA.

6.3.1 PDP

Only one module is responsible to perform tasks of PDP, in the current architecture. We denote it as "XACML PDP" in the Figure 6.1. This module is deployed as a part of two components. Particularly, DMS and TUA.

XACML PDP deployed as a part of the DMS, is responsible to evaluate authorization requests and respond decisions, during data upload and download process. Evaluation of the authorization requests are performed against server-side policies, using libraries provided by Sun Microsystems.

Sun's XACML Implementation is an access control policy evaluation engine written entirely in the Java programming language. It supports XACML 1.x and 2.0 core features, and provides a number of powerful extensions and extensibility points [35].

XACML PDP listens on the certain port to receive an authorization requests. Port number is configurable and depends on the IT infrastructure of the organization. Note that, XACML does not specify the transmission mechanisms of the authorization messages, therefore we was free to choose appropriate mechanism, by also considering time constraints. In particular, we chose to exchange authorization messages via network sockets between PEP and PDP.

XACML PDP deployed on TUA, is responsible to evaluate authorization requests and respond decisions, during data operations performed outside cloud. Evaluation of the authorization requests are performed against data-centric policies, using aforementioned libraries provided by Sun Microsystems. Communication with this module is possible only by internal procedure calls.

6.3.2 PEP

Similarly to PDP, there is only one main module responsible to perform tasks of PEP, in the current architecture. We denote it as "XACML PEP" in the Figure 6.1. This module is deployed as a part of two components. Particularly, Cloud-client application and TUA.

XACML PEP deployed as a part of Cloud-client application, acts as a bridge between users and cloud data sharing system, during data upload/download process. It generates authorization requests, and enforces authorization decisions made by XACML PDP deployed in DMS. Authorization requests may be addressed to both, KMS and DMS, during key and data operations accordingly.

To generate an authorization request XACML PEP adds values, such as user credentials and action to be performed to the existing request template, and sends it over the network. Connection between PEP and PDP is maintained over socket connection, on the preconfigured port.

XACML PEP deployed on TUA, is responsible to request local XACML PDP, and enforce its decisions. For example, enable or disable data sharing based on the decision. This interaction is implemented by internal procedure calls between them. Request generation process is the same for all XACML PEPs.

6.3.3 Authorization Proxy

Additonally we would like to note "XACML Proxy" module, which is implemented as a part of the KMS. XACML Proxy forwards authorization requests and responses between Cloud-client application and DMS, before key registration and retrieval operations. This module is implemented to support design decision made in Section 5.1.4.

XACML Proxy listens on the certain port to receive an authorization requests. After receiving authorization request, it forwards received request to the XACML PDP, over the newly initiated socket connection. Authorization decision traverses the same route in reverse order to reach the requester. Again, the port number is configurable and depends on the IT infrastructure of the organization.

6.4 Data Management

Data management is mainly done by three components in the current architecture. Particularly, these components are DMS, Cloud-client application and TUA.

6.4.1 Data Upload

Four modules are involved to the data upload process in the current architecture.

CDMI-Server

We have compared existing open source CDMI servers and chose the reference implementation developed by SNIA [2], based on certain criteria (Appendix B). CDMI-Server module is deployed as a part of the DMS. Note that, reference implementation does not support all functionality described in CDMI v.1.0.x specification, such as queuing capability, that is critical while handling concurrent data management operations.

CDMI-Server is a web service developed in Java, performing server operations in accordance with CDMI specification. It handles RESTful data upload and download operations in our implementation architecture. Additionaly, it handles creation and modification of all types of metadata (see Section 4.3), including vendor-specific metadata. Latest version at the time of implementing current architecture was *1.0.d*. It was succesfully deployed on GlassFish application server v4.0 [9].

CDMI-Server creates metadata file for each uploaded data file automatically. Note that, CDMI-Server does not serve files without metadata. In other words, data file should have metadata file in the same folder, in order to be downloadable.

Operations Module

We have developed the "Operations Module" (OM), that is deployed as a part of the DMS. It performs integrity checks and updates the PolicyMapping file as described at the end of the Section 5.6.1.

SNIA announces new reference implementations of CDMI-Server in a periodic intervals. To achieve better interoperability with future releases and to simplify deployment efforts, OM is developed as a separate module and its functionality is not integrated to the CDMI-Server.

OM monitors specific containers in DMS and reacts on file creation events. When data and its metadata file are uploaded to the DMS, OM extracts keyID and PVM values from the metadata file. Then, it checks if file with keyID name exists in the predefined folder. If file does not exist, OM deletes data file and its metadata file immediatelly (data upload operation fails). Otherwise, OM compares PVM value contained in the file with PVM value in the metadata. If values are the same, OM extracts the name of data-centric policy file from the metadata, if applicable. Then it takes a hash of the policy file and adds it as "policyreference" value of the metadata. Later, OM checks if the given hash exists in the PolicyMapping file. If hash exists, OM deletes the policy file, since PRP is already contains the same policy. If hash does not exist, OM keeps a policy file in the PRP. Finally, OM deletes PVM field and its value from the metadata, and deletes file with keyID name(containing PVM value).

SHA-256 was used as a hash function, to calculate the value of "policyreference" field. Collision resistance property of SHA-256 is important in providing uniqueness of the "policyreference" value. Note that collisions might occur with negligibly small probability.

Currently, queues are not implemented in the CDMI-Server implementation. Therefore, large number of concurrent data file creations might occur in a real production environment. Even with multithreading capabilities, OM was not able to handle concurrent data file creations and their concurrent processing in this case. Therefore, we have implemented artificial queuing which resolves this problem.

In detail, OM creates an empty file with the name of the uploaded file in a "queuing" folder. This is done by the multithreading capabilities of Java. Concurrently, OM obtains list of file in the "queuing" folder and processes it. File names in the list are used to process uploaded files in an ordered manner. After each processed file, OM deletes corresponding empty file from "queuing" folder.

Note that maximum number of concurrent data file creations is determined by the configuration of the web server, and not by CDMI-Server.

CDMI-Client on Cloud-client application

CDMI-Client module is deployed as a part of Cloud-client application and TUA. Data upload operation involves only CDMI-Client deployed on Cloud-client application. CDMI-Client is responsible for data operations such as data download/upload in the current implementation.

CDMI-Client first checks if data file is already exists at the destination, before uploading data file to the DMS. In typical scenarios this check is done by executing *HTTP HEAD* request method. However, current reference implementation does not support this method. As an alternative, we use *HTTP GET* request to obtain *HTTP* response headers, without subsequent download of the data. It is possible to identify if data file exists in the server by examining response status. Moreover, this check is done after data upload, to verify succesful storage of the file after integrity checks (Section 5.6.1).

CDMI-Client encrypts data with Advanced Encryption Standard (AES) cipher algorithm in Cipher Block Chaining (CBC) mode [46], before uploading it to the DMS. We use IHE DEN [8] class library to perform encryption process. This library have been developed by Philips. Encryption key length is 128 bits, and it is a minimum key length supported by AES algorithm. Key is generated randomly by CDMI-Client. NIST [1] claims that security of AES-128 is sufficient at the very least up to 2030 [15].

According to the CDMI specification :
"CDMI version 1.0.1 introduces the concept of value transfer encoding to enable the storage and retrieval of arbitrary binary data via CDMI content-type operations. Data objects created by CDMI 1.0 clients through CDMI content-type operations shall have a value transfer encoding of "utf-8", and data objects created through non-CDMI content-type operations shall have a value transfer encoding of "base64"."

Since encryption results in arbitrary binary data, we encode encrypted data to Base64 format before uploading it to the DMS. Later, CDMI-Client calculates PVM value (Section 5.1.4), by taking SHA-256 hash of the encryption key in combination with keyID. Furthermore, CDMI-Client adds keyID, PVM and the name of data-centric policy file to the metadata.

Finally, data upload getting performed using *HTTP PUT* request method. HttpClient v4.2.5 libraries, provided by Apache Sotware Foundation [14] has been used to perform *HTTP* request methods.

KMIP-Server

KMIP-Server module is deployed as a part of the KMS. KMIP-Server mainly involved during the key registration and retrieval operations (Described in the section 6.5.2).

KMIP-Server involved in data upload process, by providing PVM and keyID values to the DMS, as described in Section 5.1.4. To calculate PVM, KMIP-Server first concatenates values of generated keyID(Unique Identifier for the key) and key itself. Later, it calculates SHA-256 hash over this concatenated string.

Finally, KMIP-Server uploads file with the name keyID and value PVM to the DMS, by performing *HTTP PUT* request.

6.4.2 Data Download

Three modules are involved in the data download process in current architecture. Particularly, these modules are CDMI-Server, CDMI-Client and XACML PDP. We will not describe functionality of CDMI-Server, since all its functionality is described in the previous section (Section 6.4.1).

XACML PDP

As was mentioned before, CDMI-Server reference implementation is not complete. All metadata should be transferred along with the data itself, during data download operation, according to CDMI specification. However, this functionality is not implemented yet. Additionally, CDMI-Server does not respond with the data, if metadata for the data does not exist. To overcome these shortcomings we use following technics:

- We create dummy metadata for metadata, in order to be able to download it.
- We download metadata file separately from data file.

As was mentioned before, XACML PDP is responsible to evaluate authorization requests and respond the decisions to the requester. XACML PDP creates a metadata for the metadata of the requested data, during authorization process. In particular, metadata for the metadata is created only if "Permit" decision made, after evaluation of authorization request to download data.

CDMI-Client on Cloud-client application

To download a data file and its metadata from the DMS, CDMI-Client performs two independent *HTTP GET* requests, providing the name of the data file and metadata file. After data file and its metadata is downloaded, CDMI-Client obtains keyID from the metadata. Cloud-client application obtains key from KMS, based on keyID. Key retrieval process is described in Section 6.5.2.

Then, CDMI-Client decodes Base64 encoded data file and decrypts it, using obtained key. Decryption is performed by using aforementioned IHE DEN class library developed by Philips.

6.4.3 Data Sharing

Two modules are involved in the data sharing process in the current architecture. Particularly, these modules are CDMI-Server and CDMI-Client deployed as a part of the TUA. We do not consider policy evaluation steps, since they are described earlier (see Section 6.3). Again, we will not describe functionality of CDMI-Server, since all its functionality is described in the previous section (Section 6.4.1).

CDMI-Client on TUA

Metadata is needed in order be able to share the data, as it may contain ”policyreference” value. If metadata exists CDMI-Client extracts ”policyreference” value from it. Then, CDMI-Client requests to download PolicyMapping file from the CDMI-Server, by performing *HTTP GET* request. CDMI-Client extracts policy file name corresponding to the ”policyreference” from PolicyMapping file. Note that, PolicyMapping file is implemented as a comma separated file with two columns. First column contains policy references. Policy reference is a SHA-256 hash of the data-centric policy file. Second column is an absolute path to the corresponding policy file.

Afterwards, CDMI-Client requests policy file from the CDMI-Server. Then, data sharing request is evaluated against obtained policy. If decision is ”Permit”, data is copied to the predefined shared folder. Note that sharing may be done in various ways, and currently implemented method carries symbolic meaning.

6.5 Key Management

Key management is mainly done by two components in the current architecture. Particularly, these components are KMIP-Client and KMIP-Server.

6.5.1 KMIP Parser

There was no open-source KMIP client and server implementations available at the time of this project. Therefore, we developed bare minimum KMIP parser, using Java. It facilitates editing and creation process of the KMIP messages, by providing functionality of converting KMIP messages to the human readable format and back.

6.5.2 Key Registration and Retrieval

Two modules are involved in the key registration/retrieval process in current architecture. Particularly, these modules are KMIP-Server and KMIP-client.

KMIP-Server

As was mentioned before, KMIP-Server module is deployed as a part of the KMS. It is mainly responsible for server-side key registration and retrieval operations.

KMIP-Server listens to the port number 5696 to receive key registration and retrieval requests. This port number is assigned by IANA to the KMIP. Connection between KMIP-Client and KMIP-Server is maintained over a socket connection.

KMIP-Server generates random Unique Identifier for a key, during key registration process. Afterwards, it responds this Unique Identifier to the requester. KMIP-Server recognizes messages and extracts needed values from them, based on KMIP specific patterns. In particular,we have used regular expressions to achieve this goal. In order to generate a desired KMIP messages, KMIP-Server edits specifically preprepared KMIP messages.

KMIP-Server encrypts keys before their storage and decrypts them before sending them to the requester. It uses AES cipher algorithm in CBC mode with key length of 128 bits. Encryption and decryption is performed by using aforementioned IHE DEN class library developed by Philips. The encryption key is stored in the KMS.

KMIP-Client

KMIP-Client module is deployed as a part of the Cloud-client application. It is mainly responsible for the client-side key registration and retrieval operations.

KMIP client initiates a connection to the port number 5696 on the KMS. Similarly to the KMIP-Server, KMIP-Client recognizes messages and extracts needed values from them, based on KMIP specific patterns. We have used regular expressions to achieve this goal. In order to generate a desired KMIP messages, KMIP-Client edits the specifically prepared KMIP messages.

Chapter 7

Evaluation

In this chapter we present the results of performed test cases to evaluate the performance of our implementation in a real production environment. We first discuss the methodology we chose to perform these tests. We describe the configuration of different components, including hardware configuration of utilized machines, speed of network connection, installed operating systems, etc. Finally, we present the results of performance testing, followed by discussions for each test.

7.1 Methodology

To evaluate the performance of our implementation we logged the time, needed for tasks on each component/module in the system. To do so, we calculated the time difference between starting and ending of a function/procedure. Particularly, we added logging function at the start and the end of a function/procedure, and subtracting starting time from the ending time. Logging operation takes negligible time (0,002 milliseconds), therefore we do not consider it while testing. We logged all network communications between components and all local processes and functions. We have included the name of the file in a logging messages, in order to track all related processes throughout the separate components of the system.

To have better understanding of how our system performs under various scenarios, we have used files of various sizes in our testing (see Section 7.3). Our testing includes data upload, download and sharing processes, as described in the Section 5.6.

There was a minor time difference between the same operations over the files of the same size. Therefore, we have calculated the average (arithmetic mean) value of these operations and use it while analysis, along with original values. Additionally, there were a certain challenges to track the processes related to the same file, such as authorization, key management, etc. throughout the separate components. Particularly, KMIP messages do not contain any file related information. Therefore we have logged operations by including keyID in them. Later we have replaced keyID with the corresponding file name. Mapping between keyID and file name is obtained by using metadata of the file. We automated this process by developing a simple Java application.

7.2 Environment

We have performed our testing in the Amazon EC2 cloud environment, using m1.medium instance types [3]. In the current context, instance means a virtual machine running on the Amazon EC2. Below we present the configuration of the m1.medium instance, obtained via the latest version (1.66) of the freeware application, CPU-Z [10].

Processor	
Name	Intel Xeon E5
Number of cores	1
Speed	2114Mhz
Specification	Intel(R) Xeon(R) CPU E5645-2650 0 @ 2.40GHz
Instructions sets	MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, EM64T
L1 Data cache	32 KBytes, 8-way set associative, 64-byte line size
L1 Instruction cache	32 KBytes, 8-way set associative, 64-byte line size
L2 cache	256 KBytes, 8-way set associative, 64-byte line size
L3 cache	20 MBytes, 20-way set associative, 64-byte line size
Memory	
Memory Size	3840 MBytes
Memory Frequency	102.2 MHz
OS	
OS Name	Microsoft Windows Server 2012 Standard
Version	6.2.9200 Build 9200
Network Adapter	
Product Type	Citrix PV Ethernet Adapter
Adapter Type	Ethernet 802.3
Connection Speed	800 Mbit/s
Round-trip time (ping) inside cloud	bytes = 32, time $<$ 1ms, TTL = 128
Round-trip time (ping) outside cloud (25 Mbit/s)	bytes = 32, time = 35ms, TTL = 117

Table 7.1: Configuration of the m1.medium instance in Amazon EC2.

Additionally, we measured the speed of the network connection between two m1.medium instances, by using freeware version of the LAN Speed Test (v3.4) application [56].

7.3 Results

In this section we present the results of the performance evaluation. We have calculated an average time per set of 100 files, of the same size, for both, data upload and data download

processes. We have tested 22 different file sizes, specifically {0.1, 0.5, 1, 2, 3,...,20} megabytes. We will refer to these files as the "tested file set" throughout this section. Note that tested file set contains $22 \times 100 = 2200$ files.

7.3.1 Data Upload

Below we present the scatter plots for the tasks performed while data upload, with execution time depending from the file size. Particularly we have analyzed performance while uploading file to the CDMI-Server (not including any other operations), performance of AES encryption , Base64 encoding and overall time that file upload process took for tested file set.

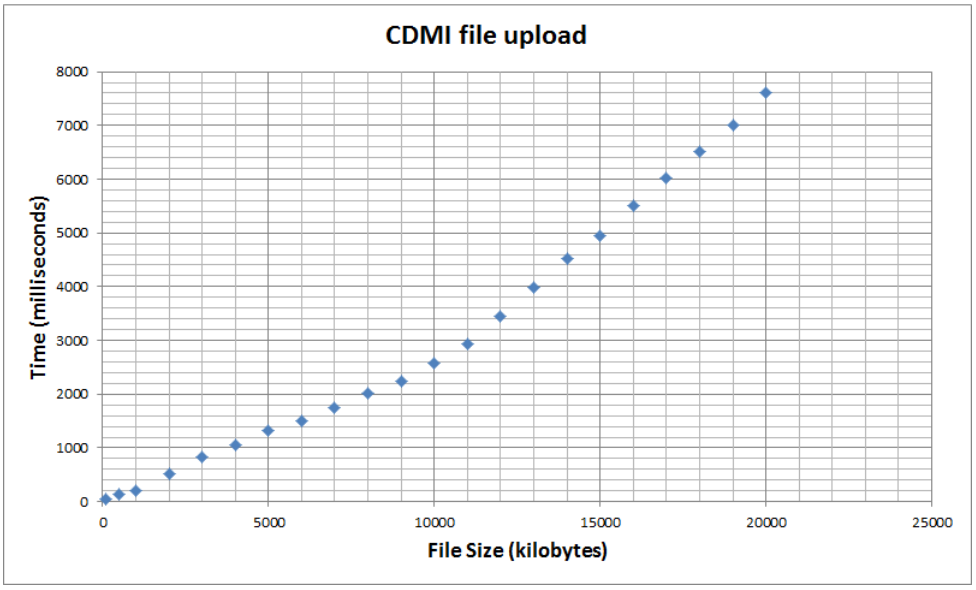

Figure 7.1: Average data upload time to the DMS.

Figure 7.1 represents average data upload times to the DMS. We have noted that time to upload each additional megabyte of data increases superlinearly for file sizes bigger than 11MB. CDMI-Server stores chunks of data in the memory, before storing it in the hard drive, until all data is received via network connection. This might lead to the overhead of managing the data in the memory by the CPU.

Figure 7.2 represents average data encryption times for the tested file set in the cloud. The ratio of the file size and the time is close to linear, which is the expected result. While data size grows, encryption algorithms handles more data, thus takes more time to process it.

Figure 7.3 represents average data encoding times for the tested file set. In particular, Base64 encoding have been used. The ratio of the file size and the time is close to linear, which is the expected result. Principally, Base64 is a simple algorithm changing the representation of the data rather than processing it.

Figure 7.4 represents average of the overall time spent to upload the file for the tested file set. Data upload time increases non-linearly. It is more significantly noticeable with the file sizes more than 11MB. We assume that it relates to the processing of the complete file in the memory while the data upload process, in both sides of the communication, which

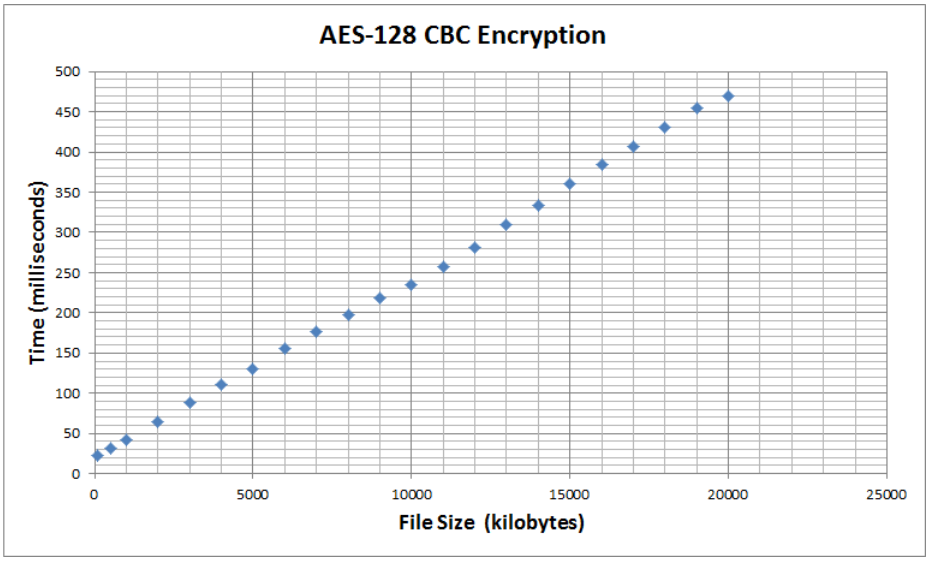

Figure 7.2: Average AES encryption performance on m1.medium instance..

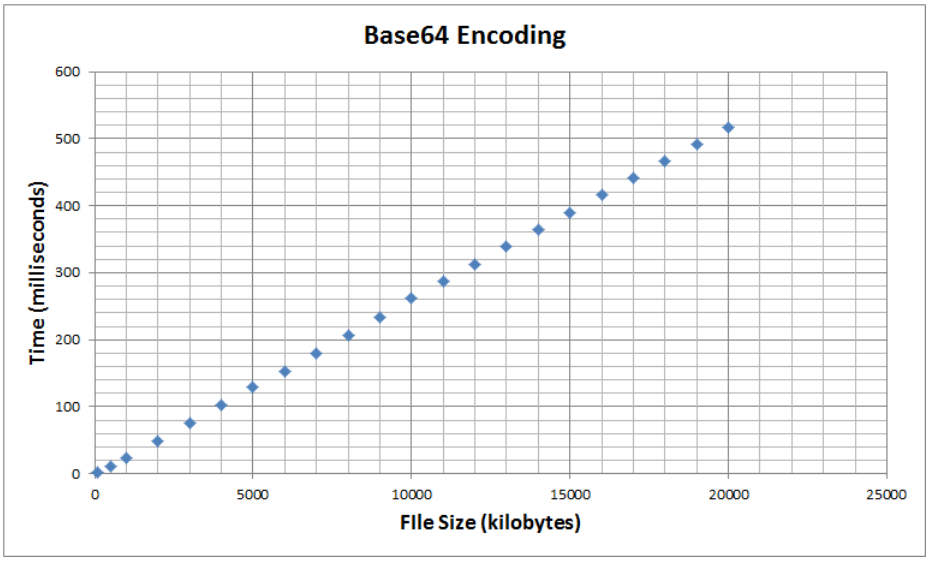

Figure 7.3: Average Base64 encoding performance on m1.medium instance..

leads to processing overhead while the data size grows. Obviously, storage of files with bigger size should take more time. However, high performance SCSI disks minimizes the read/write operations to the storage to the negligible value (approximately, 1.3 milliseconds per 1 megabyte).

Figure 7.5 represents the performance while uploading complete tested file set, without focusing on the average values. The deviation from the average time is more obvious while data size grows. We assume that it relates to the resource pooling in the cloud environment, that decreases/increases the utilized resources on demand. While no extensive data operations are performed cloud may decrease resource capacity, and then increase it back on demand. This type of resource management may increase initialization time for the process,

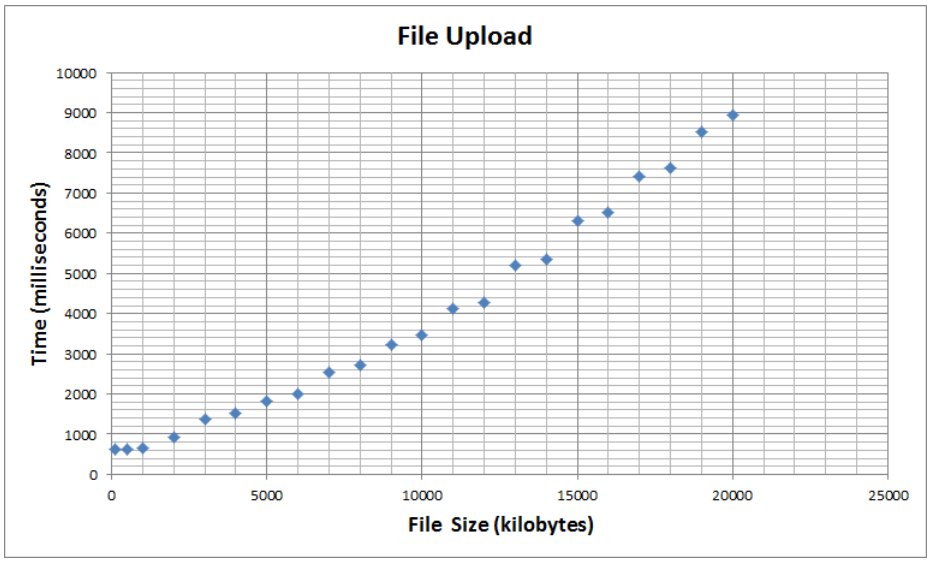

Figure 7.4: Average performance of the complete file upload process on m1.medium instance.

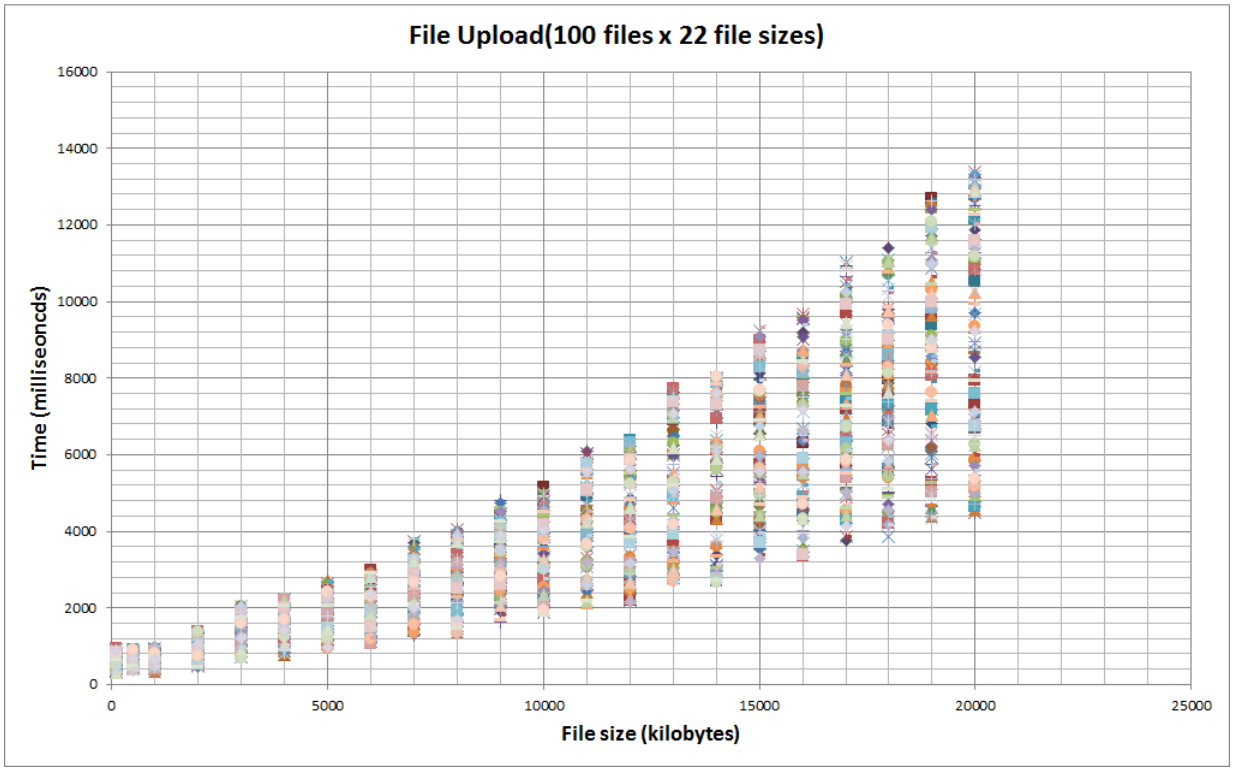

Figure 7.5: Performance of the complete file upload process on m1.medium instance.

especially on the early stages and increase overall time of execution.

Additionally we present the Table 7.2 with tasks performed while data upload, with execution time not depending from the file size. We describe the sequence of operations in the Figure 7.6 while data upload to easier track the flow of the operations.

We would like to explain in details certain sections of this table. "PVM check, Queuing,

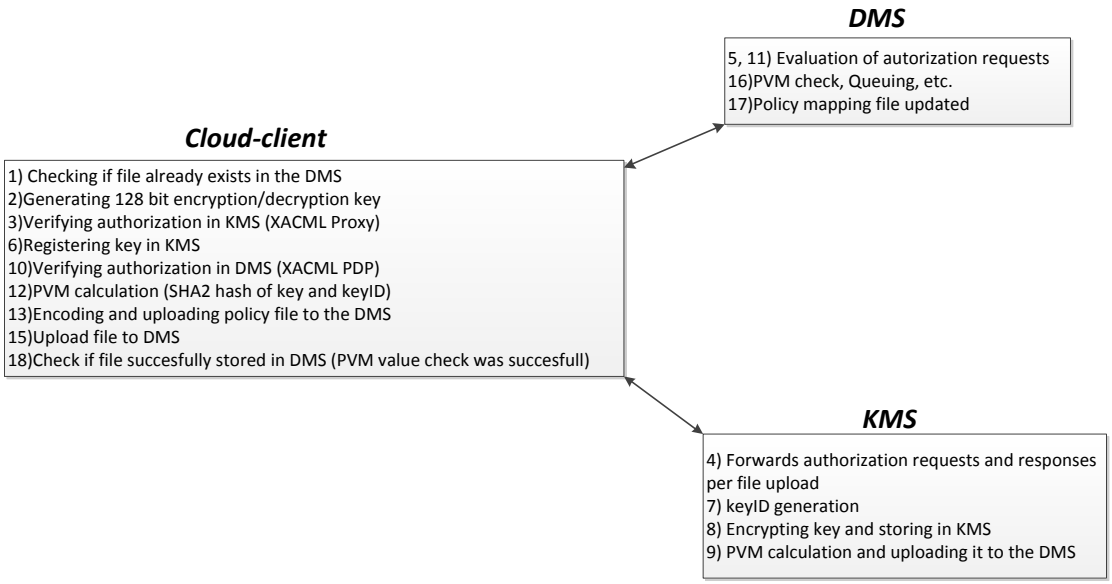

Figure 7.6: Sequence of operations while data upload.

etc." performed by the DMS and represents integrity checks and capabilities similar to queuing while network communications. "Evaluation of authorization requests" involves the processing of one policy file that consists of 4 server-side policies; two policies for key upload/-download, and two for data upload/download. Verification of the authorization via KMS (XACML Proxy) introduces certain overhead, due to the communication overhead of forwarding messages between Cloud-client application and DMS. "Registering key in KMS" time represents only KMIP message exchange, without including the authorization procedures.

Additionally, we describe the relative timing of the operations that do not depend on the processed file size, while data upload, in the Figure 7.7. Obviously, verifying authorization in KMS (XACML Proxy) while key upload operation introduces latency.

Operation while Upload	*Average Time (milliseconds)*
Operations Module on DMS	
PolicyMapping file updated	4
PVM check, Queuing, etc.	24
XACML PDP on DMS	
Evaluation of authorization requests	58
XACML PEP on Cloud-Client	
Verifying authorization in DMS (XACML PDP)	41
Verifying authorization in KMS (XACML Proxy)	205
CDMI-client on Cloud-Client	
Checking if file already exists in the DMS	62
Encoding and Uploading policy fle to the DMS	41
Check if file succesfully stored in DMS (PVM value check was succesfull)	111
Generating 128 bit encryption/decryption key	0.07
PVM calculation (SHA2 hash of key and keyID)	0.2
KMIP-client on Cloud-Client	
Registering key in KMS	71
KMIP-Server on KMS	
PVM calculation and uploading it to the DMS	52
Encrypting key and storing in KMS	2
keyID generation	0.2
XACML Proxy on KMS	
Forwards authorization requests and responses per file upload	195

Table 7.2: Execution time of the tasks, while data upload, not depending from the file size.

7.3.2 Data Download

Below we present the scatter plots for the tasks performed while data download, with execution time depending from the file size. Particularly we have analyzed performance while downloading file from the CDMI-Server (not including any other operations) to the Cloud-Client application, performance of AES decryption , Base64 decoding and overall time that file download process took for tested file set.

Figure 7.8 represents average data download times from the DMS. We have noted that

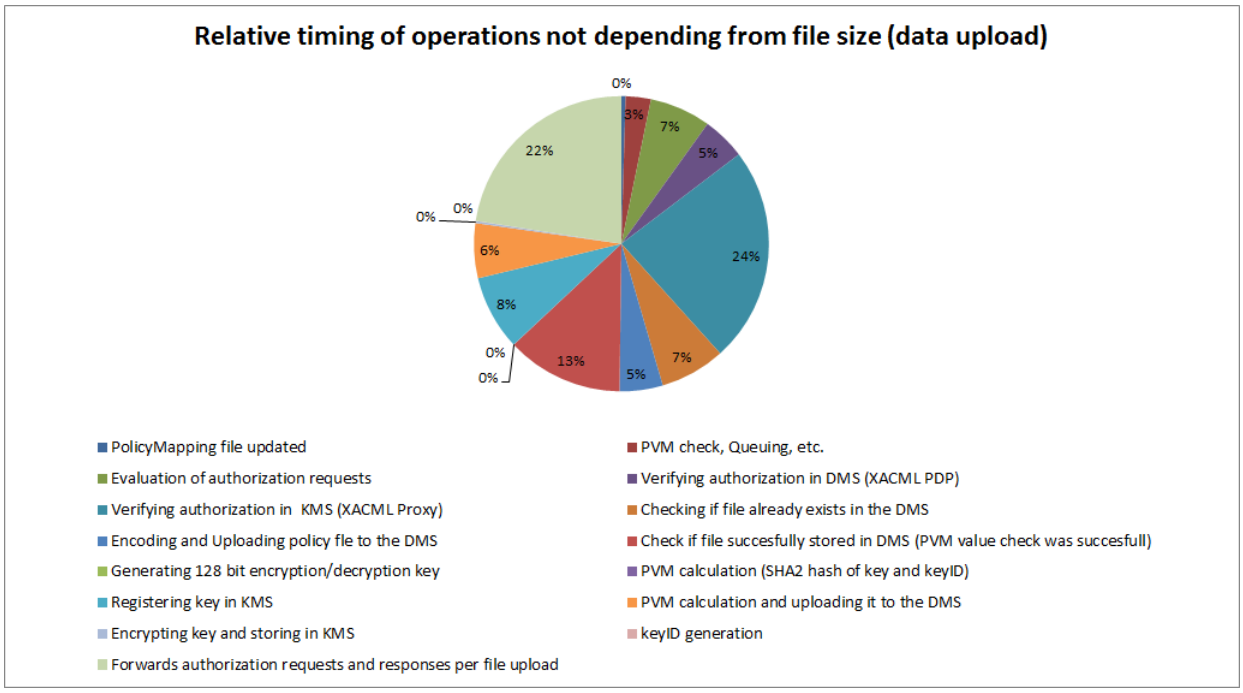

Figure 7.7: Relative timing of operations not depending from file size while data upload.

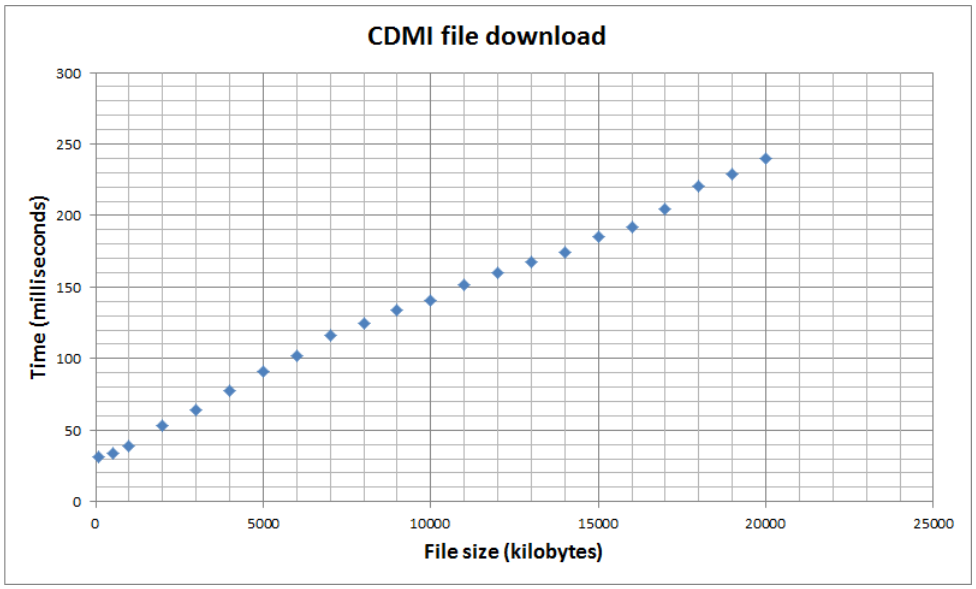

Figure 7.8: Average data download time to the DMS.

time to download each additional megabyte of data is increases non linearly for file sizes smaller than 7MB and bigger than 10MB. Again, we assume that it might be related to the specifics of the cloud environment, in particular to the dynamic resource pooling features, and storage of the big data files in the memory until their complete receiving. However, it is obvious that files with size between 7-10MB can be downloaded more efficiently, since data download times increase sublinearly between 7-10MB file sizes.

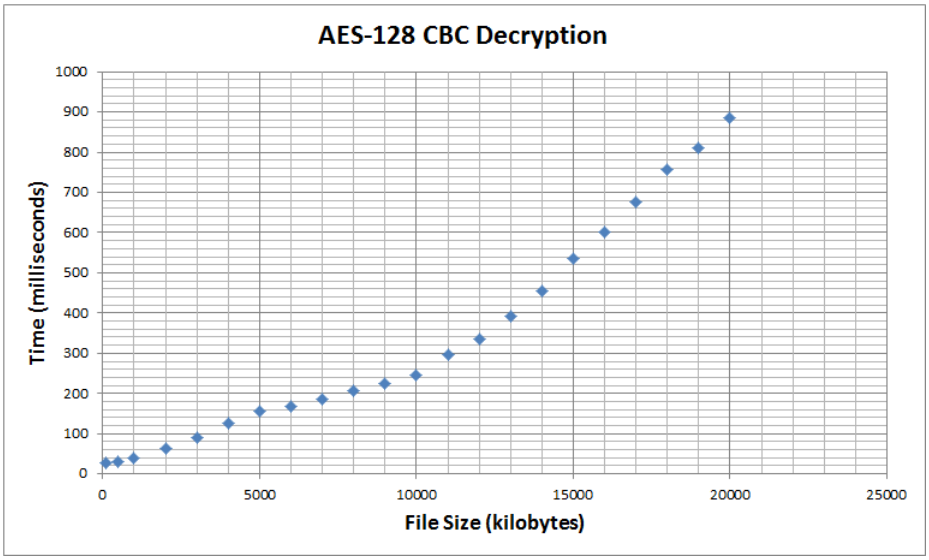

Figure 7.9: Average AES decryption performance on m1.medium instance..

Figure 7.9 represents average data decryption times for the tested file set in the cloud. The ratio of the file size and the time is far from linear, which is not expected result. Decryption is performed in sublinear time until the file size reaches to 10MB. In fact decryption times for the file sizes of 5MB an 6MB are very close. However, decryption time grows superlinearly for the files bigger that 10MB. It might be related to the CPU's cache capacity. Since all uploaded data has similar structure, certain caching mechanisms might be leveraged to optimize the decryption process (until certain file size).

Figure 7.10 represents the performance while decrypting complete tested file set, without focusing on the average values. Again the deviation from the average time is more obvious while data size grows. We assume that it relates to the resource pooling in the cloud environment, that decreases/increases the CPU frequency on demand, and store or release certain initialization data in the CPU caches.

Figure 7.3 represents average data decoding times for the tested file set. The ratio of the file size and the time is close to linear, which is the expected result. We assume that resource pooling slightly affects Base64 decoding and deviates the plot from the linearity at multiple points.

Figure 7.12 represents average of the overall time spent to download the file for the tested file set. Data download time is close to linear, which is the expected result. However, there are certain deviations, that can be justified by ubiquitous network jitters.

Figure 7.13 represents the performance while downloading complete tested file set, without focusing on the average values. Againg, the deviation from the average time is more obvious while data size grows. We assume that the reason is the same as was mentioned before.

We present the Table 7.3 with tasks performed while data download, with execution time not depending from the file size. We describe the sequence of operations in the Figure 7.14 while data download to easier track the flow of the operations.

We would like to explain in details certain sections of this table. "Download metadata file from DMS" performed separately, since current reference implementation of CDMI does not reply metadata automaticaly with the data. This introduces certain overhead. "Decrypting

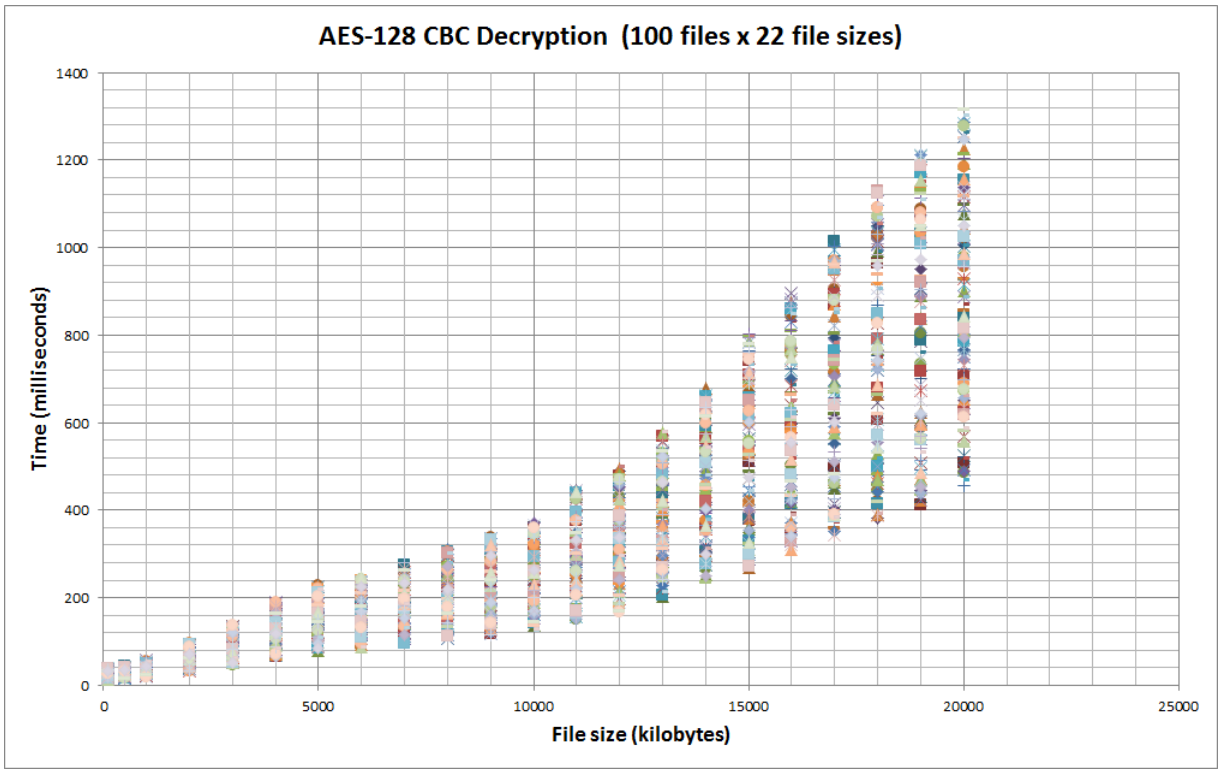

Figure 7.10: AES decryption performance on m1.medium instance..

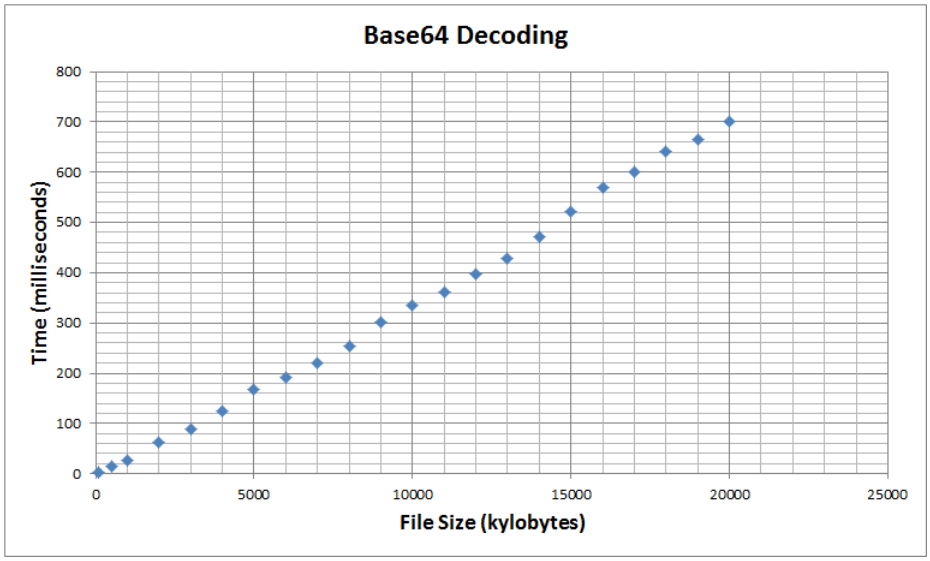

Figure 7.11: Average Base64 decoding performance on m1.medium instance..

stored key before sending it to the Cloud-client" includes time of network communication. "Retrieveing key in KMS" time represents only KMIP message exchange, withoud including the authorization time.

Additionally, we describe the relative timing of the operations that do not depend on the processed file size, while data download in the Figure 7.15. Obviously, verifying authoriza-

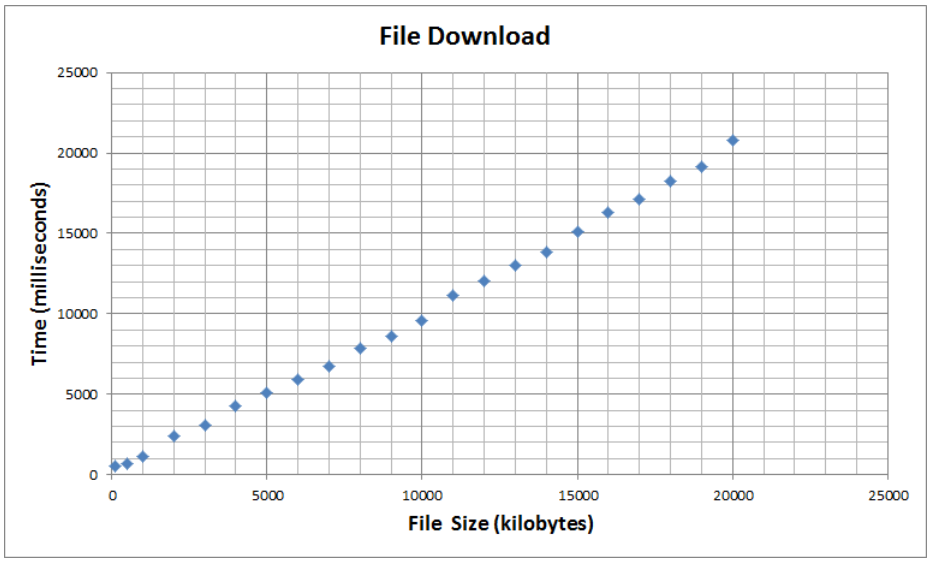

Figure 7.12: Average performance of the complete file download process on m1.medium instance..

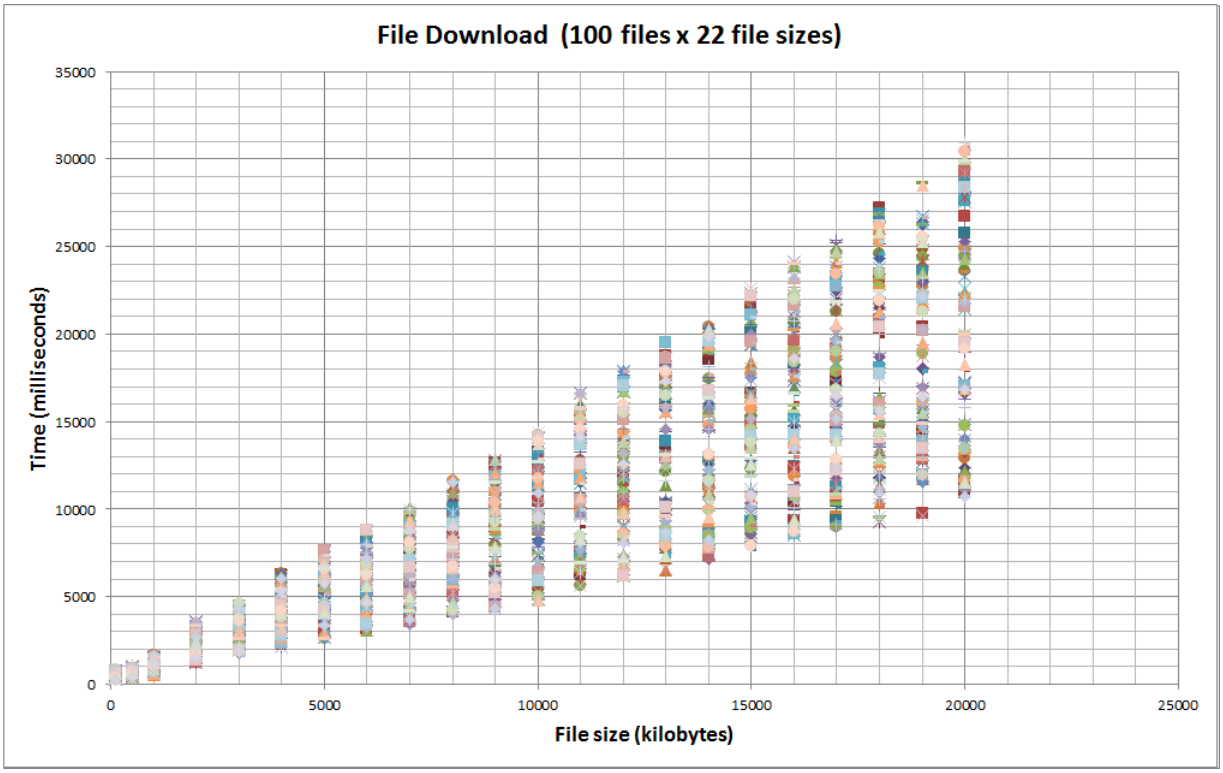

Figure 7.13: Performance of the complete file download process on m1.medium instance.

tion in KMS (XACML Proxy) while key upload operation introduces latency.

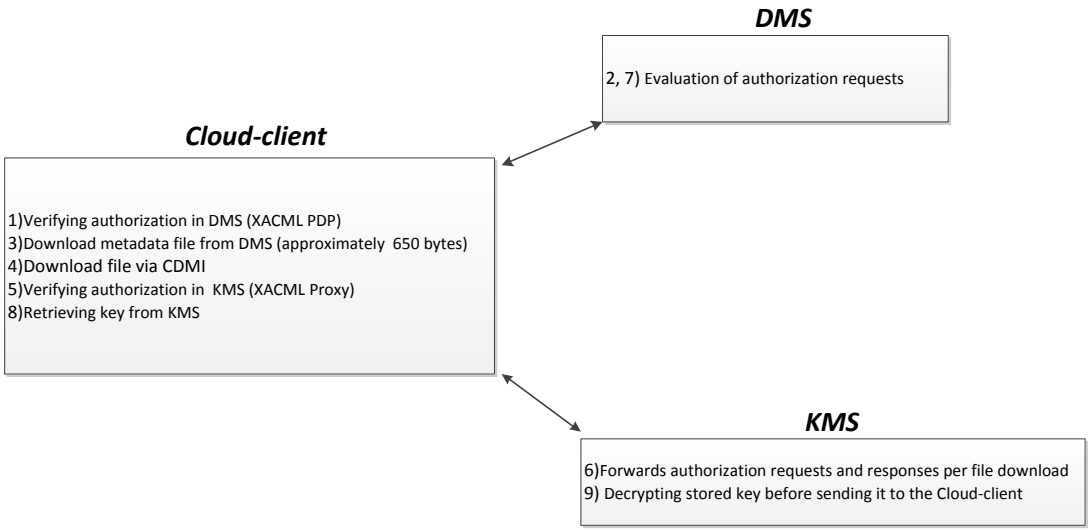

Figure 7.14: Sequence of operations while data download.

Operation while Download	*Average Time (milliseconds)*
XACML PDP on DMS	
Evaluation of authorization requests	58
XACML PEP on Cloud-Client	
Verifying authorization in DMS (XACML PDP)	97
Verifying authorization in KMS (XACML Proxy)	200
CDMI-client on Cloud-Client	
Download metadata file from DMS (approximately 650 bytes)	35
KMIP-client on Cloud-Client	
Retrieving key from KMS	40
KMIP-Server on KMS	
Decrypting stored key before sending it to the Cloud-client	1.8
XACML Proxy on KMS	
Forwards authorization requests and responses per file download	195

Table 7.3: Execution time of the tasks, while data download, not depending from the file size.

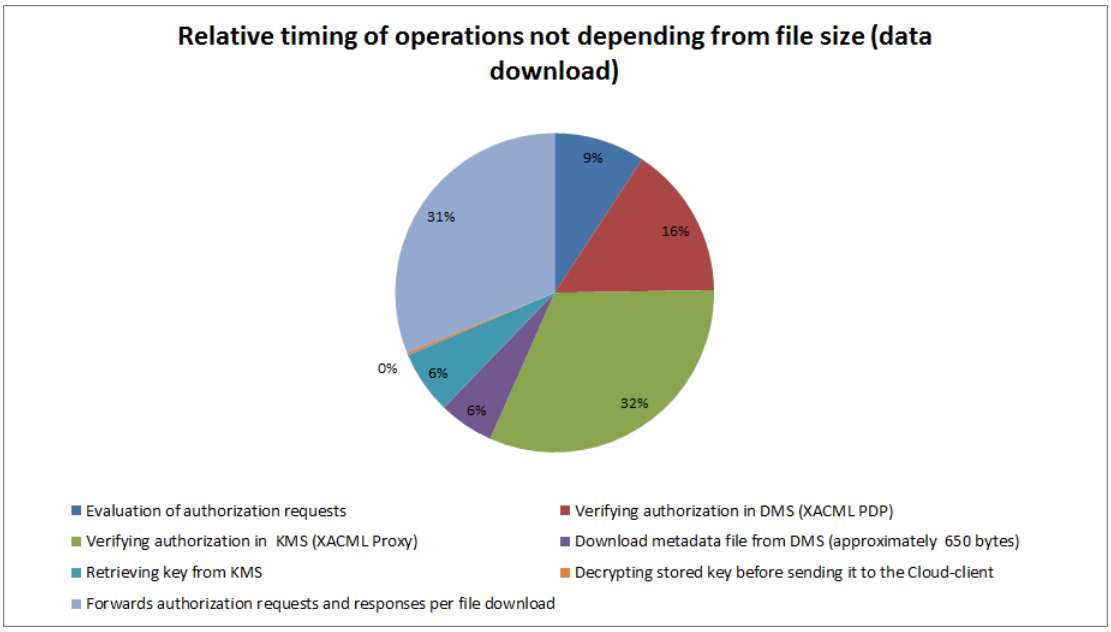

Figure 7.15: Relative timing of operations not depending from file size while data download.

7.3.3 Data Sharing

Below we present two scatter plots for the tasks performed while data sharing, with execution time not depending from the file size, but from the size of the policies. Particularly we have analyzed performance while retrieving policy file from the CDMI-Server and performance of evaluating policy agains data sharing requests. We performed 100 tests for each task. Policies was downloaded to the local machine outside cloud via TUA and evaluated. Policies contain only one rule. Average size for the policy is 1869 bytes, in our tests.

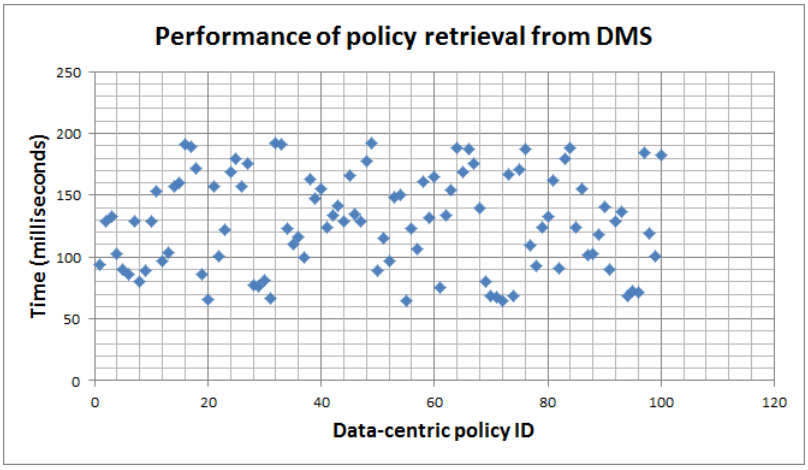

Figure 7.16: Performance of policy retrieval from DMS.

Figure 7.16 represents 100 policy retrieval operations. There is a certain deviation from average time for retrieving policy. We have calculated an average of time for policy retrieval as 128 milliseconds utilizing the 25Mbit/s Internet connection speed.

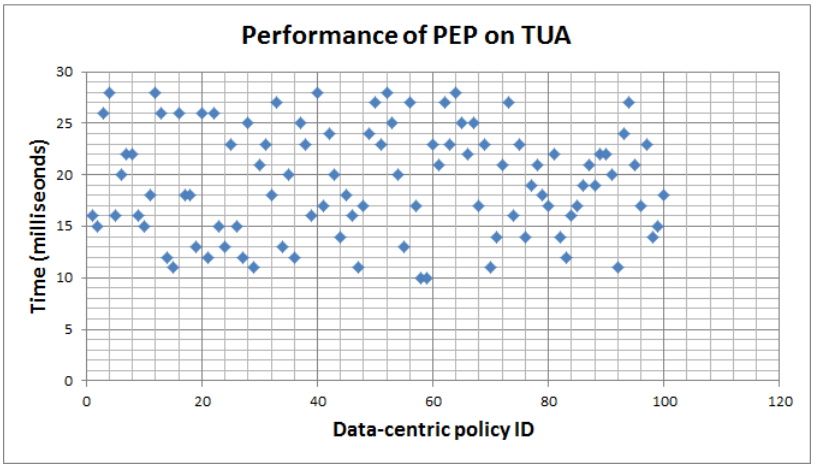

Figure 7.17: Performance of policy evaluation on local machine (TUA).

Figure 7.17 represents the policy evaluation performance on the machine with Intel Core 2 Duo T5800 processor (2.0GHz), with 4GB of pre-installed memory, and Serial ATA hard disk drive (5400rpm). Time to evaluate request against the policies of aforementioned size (1869 bytes) took 19 milliseconds in average, which is negligible result in our opinion.

7.4 Discussions

We present performance testing results of our implementation, deployed in the Amazon EC2. Our analysis shows that cloud environment might perform unstable in some cases (e.g. Figure 7.9), due to the multinentant environment that benefits from flexible resource pooling. However, most of the operations are performed significantly fast. Due to the abstraction that cloud provides it was not possible to measure all the characteristics of the underlying infrastructure, such as speed of HDD (cloud utilizes network attached storage arrays). However, we assume that performance bottlenecks are CPU of the cloud machines, that handles memory access, read and write operations from/to the cloud storage. Amazon EC2 tend to decrease the CPU frequency on demand, to provide flexible resource sharing and cost effectiveness in its multinent environment.

Performance of our implementation may be increased by stabilizing allocated resources by Amazon EC2. Additionally, performance can be slightly increased by removing XACML Proxy module from the KMS and verifying authorization for the key registration directly on XACML PDP, deployed on DMS. Approximately, XACML Proxy introduces overhead of 100 milliseconds.

Chapter 8

Conclusions and Future work

In this chapter, we present the conclusion of the work performed for this thesis. Moreover we propose paths for future work in the direction of designing and implementing secure cloud-based systems.

8.1 Conclusions

In this thesis, we presented an approach for designing and implementing access control policies for the cloud-based systems. In particular, we have analyzed existing work in this field and propose a system architecture, based on our analysis. Furthermore, we have constructed a prototype to validate our architecture. Our main goal was to determine the feasibility of implementing secure data sharing system that will enforce policy based access control, utilize benefits of cloud and will comply with existing standards and specifications. We have answered all four research questions presented at the beginning of this thesis.

To answer the first research question we have performed the analysis of existing work, which can be divided into three parts. First, we have defined security requirements for the data stored in cloud. Based on these requirements we have proposed certain characteristics for an access control system in the cloud. Finally, we have analyzed existing work in the area of designing and implementing secure data storage, key management, policy enforcement strategies and access control systems in the cloud.

To answer the second research question we have proposed the architecture of the system, providing police based access to the data, based on our analysis and intention to map theoretic requirements to the existing practical implementations. Data protection inside and outside cloud was the main objective of the proposed architecture. Our system architecture and implementation was influenced by three main standards and specifications.

We have chosen XACML as an access control policy language. We have used libraries provided by Sun Microsystems to implement access control policy evaluation engine, supporting core features of XACML 1.x and XACML 2.0.

Key management has been done in accordance with KMIP specification. There was no open source implementation available, therefore we have developed bare-minimum functionality to validate our architecture.

We chose CDMI as a data management standard in the cloud. We have analyzed existing implementations of CDMI Server and chose the reference implementation of CDMI Server provided by SNIA.

By successfully designing the architecture of the system we have prooved the possiblity of designing a cloud data management system, utilizing the benefits of cloud storage, while ensuring data security and fine-grained data access.

To answer the third research question we have implemented the proposed architecture using aforementioned standards, and deployed it in a cloud environment. Number of challenges and insufficiencies in existing implementations has been overcame in implementation phase. All components in the system have been implemented using Java programming language, therefore can be considered platform independent. By implementing the proposed architecture we have prooved the feasibility of the development and deployment of policy based access control systems in the cloud, based on existing standards and technologies.

Finally, performance of the system has been tested on Amazon EC2 and results has been analyzed, thereby answering the forth research question. We believe that our results provide a real idea about performance expectations from cloud, since testing has been performed in the cloud itself.

Our research shows that providing policy based and scalable access to the data in untrusted environment, such as cloud, is a complex task and requires interoperability between multiple components.

Furthermore, our invetigation determines that only few of the proposed access control models for cloud has been implemented in a real production environment. Most of the secure data management approaches for the cloud are exist as a prototypes and not developed further, or utilized in a scope of any one specific project.

The reason for that is a complexity and challenges introduced while mapping proposed access control schemes to the technology, in a standardized way. Standardized implementations have more business value and support, and may be rationally extended without losing backward compatibility with other systems developed according to the same standards.

Overall, it is safe to say that we have reached our goals and succeeded in the implementation of policy based access control system in the cloud.

8.2 Future Work

Our research indicated that implementing "secure" cloud-based systems is not a trivial task, since presumably requirements may vary from case to case. Moreover, insufficiencies in currently existing standards and implementations make this task even more challenging. We propose several directions for future work, with regard to our system only, since it is exhausting to propose potential directions for other cloud-based systems, due to the diversity of use case scenarios.

First, implementations corresponding to the specifications and provided by standards developing organizations (SDO) should be improved, to provide support for main functions described in the latest version of specifications., such as queuing functionality, data update, automatic retrieval of metadata, partial retrieval of the metadata.

Authentication should be added to the current system implementation, in accordance with minimum requirements mentioned in Section 5.2.

XACML provides human-readable and verbose enough policies to easy administration, not only by administrators, but also by nontechnical managers. Even more simplified policy management may be considered, by building a tool to translate authorization logic described with even more simple human-readable rules into the XACML policies.

Designing policy based access control system supporting delegation of responsibilities might be a challenge, since policies should support relations between distinct entities.

Appendix A

Case Study: Emergency Medical Service

In this section we propose a use case scenario to show the practicality of our solution in a real life. We introduce a modernization of emergency medical service to increase the quality and effectiveness, based on our prototype. We do not describe the sequence of operations, including policy enforcement, key management, data processing, etc. ,since general process flow is as described in the Chapter 6. However, we will provide certain specific implementation details to the reader.

A.1 Use Case

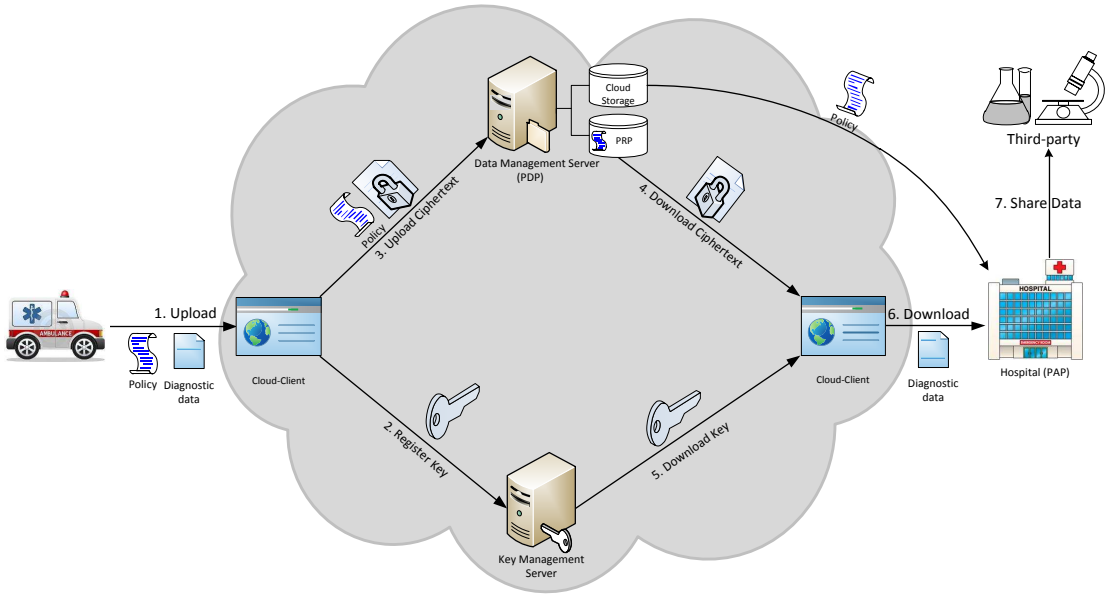

Figure A.1: Use case scenario for an emergency medical service.

Use case is described in Figure A.1. While patient is in ambulance, certain diagnostic data is sent to the cloud storage via Cloud-client application. Data is stored in specific con-

tainer(e.g. "Cardiology") according to specific case. Metadata may contain relevant information for the case, such as case criticality level, short description of the case, estimated arrival time to the hospital, etc.

Medical staff in hospital can retrieve the information about patient through the Cloud-client and prepare necessary equipment, medications, etc. in advance to immediate treatment of the patient, after he/she will arrive to the hospital. This process may be automated by developing tools to monitor the cloud containers for newly arrived cases, from hospital. CDMI provides the capability to obtain the list all files in the container by a single request. Monitoring tool may obtain the list of file indirectly, through the Cloud-client, or directly requesting CDMI-Server, depending on the regulatory policies.

This time saving technique will enable the possibility to assess patients medical records before patient arrives to hospital, which might result in more precise diagnose and subsequently in more correct treatment.

We consider a third-pary organization, conducting research(e.g. translational research [52, 47]) on certain clinical data, provided by various medical domains. Hospital may share clinical data of the patients with third-party based on data-centric policies specified for the data.

We propose that data-centric policies should be specified in the ambulance,before sending data to the cloud storage, based on existing legislation, hospital regulations, patient consent(if applicable) and other circumstances.

Policy Administration Point (PAP) tasks may be performed by hospital in current use case. PAP acts as a policy management point in the infrastructure. In particular, server-side policies may be specified by the employees(e.g. system administrators) of hospital.

Appendix B

Available CDMI Servers

We have compared available open source CDMI-Server implementations. We summarized most important features of these servers in the Table B.1 .

	CDMI Proxy (Venus-C)	*CDMI-Serv*	*CDMI for OpenStack Swift*	*SNIA CDMI Reference Implementation*
Documentation	Yes	No	Yes	Yes
Compatibility	CDMI v1.0.1	CDMI v1.0	CDMI v1.0.1	CDMI v1.0.1h
Service Type	Software Service	Software Service	Infrastructure as a Service	Software Service
Product Version	v1.0	v1.0	v1.0	1.0c
Programming Language	Python	Python	Python	Java
Licence	Apache 2	GPL	Apache 2	* Copyright (c) 2010, Sun Microsystems, Inc. *Copyright (c) 2010, The Storage Networking Industry Association.
Logging	Yes	No	Yes	No
Backend Support	Local disk, AWS S3, Azure Blob.	NA	Many different hypervisors, Block storages, Standard hardware.	NA
Platform Support	Multiplatform	Multiplatform	Linux	Multiplatform
Development Status	In development. Last Commit: September 2012	Last Commit: March 2011	In development Last Commit: February 2013	In development.
Quality of Code	Good. Moderate comments through the code.	Good.Very few comments in the code.	Good. A lot of comments through the code.	Good. Not much comments in code.
Score	7/10	3/10	8/10	9/10

Table B.1: Comparison of CDMI implementations.

Appendix C

KMIP Messages

C.1 Key registration request.

```
42007801000001104200770100000038420069010000002042006
a020000000400000001000000042006b02000000040000000100
00000042000d020000000400000001000000042000f01000000c
842005c050000000400000003000000042007901000000b04200
570500000004000000020000000042009101000000384200081 0
000003042000a0700000018437279707 46f677261706869632055
73616765204d61736b42000b020000000400000 0c00000000420
08f0100000058420040010000005042004205000000040000000 1
0000000042004501000000184200430800000010012345678 9abc
def0123456789abcdef42002805000000040000000300000000042
002a020000000400000080000000 0
```

XML representation.

```
<RequestMessage\>
  <RequestHeader>
    <ProtocolVersion>
      <ProtocolVersionMajor type="Integer" value="1"/>
      <ProtocolVersionMinor type="Integer" value="1"/>
    </ProtocolVersion>
    <BatchCount type="Integer" value="1"/>
  </RequestHeader>
  <BatchItem>
    <Operation type="Enumeration" value="Register"/>
    <RequestPayload>
      <ObjectType type="Enumeration" value="SymmetricKey"/>
      <TemplateAttribute>
        <Attribute>
          <AttributeName type="TextString" value="Cryptographic Usage Mask"/>
          <AttributeValue type="Integer" value="Decrypt Encrypt"/>
        </Attribute>
      </TemplateAttribute>
      <SymmetricKey>
        <KeyBlock>
          <KeyFormatType type="Enumeration" value="Raw"/>
          <KeyValue>
            <KeyMaterial type="ByteString" value="0123456789abcdef0123456789abcdef"/>
          </KeyValue>
          <CryptographicAlgorithm type="Enumeration" value="AES"/>
          <CryptographicLength type="Integer" value="128"/>
        </KeyBlock>
      </SymmetricKey>
    </RequestPayload>
  </BatchItem>
```

```
</RequestMessage>
```

C.2 Key registration response.

```
42007b01000000b042007a0100000048420069010000002042006
a0200000004000000010000000042006b02000000040000000100
0000042009209000000080000000 4f9a54e942000d020000000
4000000010000000042000f010000005842005c05000000040000
00030000000042007f0500000004000000000000000042007c010
0000030420094070000002433653236323961372d386238322d34
6339352d393235382d346664366536626139366334000000 00
```

XML representation.

```
<ResponseMessage>
  <ResponseHeader>
    <ProtocolVersion>
      <ProtocolVersionMajor type="Integer" value="1"/>
      <ProtocolVersionMinor type="Integer" value="1"/>
    </ProtocolVersion>
    <TimeStamp type="DateTime" value="2012-04-27T08:12:25+00:00"/>
    <BatchCount type="Integer" value="1"/>
  </ResponseHeader>
  <BatchItem>
    <Operation type="Enumeration" value="Register"/>
    <ResultStatus type="Enumeration" value="Success"/>
    <ResponsePayload>
      <UniqueIdentifier type="TextString" value="3e2629a7-8b82-4c95-9258-4fd6e6ba96c4"/>
    </ResponsePayload>
  </BatchItem>
</ResponseMessage>
```

C.3 Key retrieval request.

```
42007801000000904200770100000038420069010000002042006
a0200000004000000010000000042006b02000000040000000100
00000042000d020000000400000001000000042000f010000004
842005c05000000040000000a000000004200790100000030420 0
9407000000243635623134383166 2d336633612d343537662d396
261392d626236663638313462653730000000 00
```

XML representation.

```
<RequestMessage>
  <RequestHeader>
    <ProtocolVersion>
      <ProtocolVersionMajor type="Integer" value="1"/>
      <ProtocolVersionMinor type="Integer" value="1"/>
    </ProtocolVersion>
```

```
    <BatchCount type="Integer" value="1"/>
  </RequestHeader>
  <BatchItem>
    <Operation type="Enumeration" value="Get"/>
    <RequestPayload>
      <UniqueIdentifier type="TextString" value="65b1481f-3f3a-457f-9ba9-bb6f6814be70"/>
    </RequestPayload>
  </BatchItem>
</RequestMessage>
```

C.4 Key retrieval response.

```
42007801000000904200770100000038420069010000002042006
42007B010000012042007A01000000484200690100000020420 06
A020000000400000001000000042006B02000000040000000100
0000004200920900000008000000004F9A54E742000D020000000
40000000100000000420 00F01000000c842005C05000000040000
000A0000000042007F0500000004000000000000000042007C010
00000a042005705000000040000000200000004200940700000 0
2434396131636138382D366265612D346662322D623435302D376
535383830326333303338000000042008F010000005842004001
0000005042004205000000040000000100000004200450100000
018420043080000001000112233445566778899aabbccddeeff42
002805000000040000003000000042002A02000000400000008
000000000
```

XML representation.

```
<ResponseMessage>
  <ResponseHeader>
    <ProtocolVersion>
      <ProtocolVersionMajor type="Integer" value="1"/>
      <ProtocolVersionMinor type="Integer" value="1"/>
    </ProtocolVersion>
    <TimeStamp type="DateTime" value="2012-04-27T08:12:23+00:00"/>
    <BatchCount type="Integer" value="1"/>
  </ResponseHeader>
  <BatchItem>
    <Operation type="Enumeration" value="Get"/>
    <ResultStatus type="Enumeration" value="Success"/>
    <ResponsePayload>
      <ObjectType type="Enumeration" value="SymmetricKey"/>
      <UniqueIdentifier type="TextString" value="49a1ca88-6bea-4fb2-b450-7e58802c3038"/>
      <SymmetricKey>
        <KeyBlock>
          <KeyFormatType type="Enumeration" value="Raw"/>
          <KeyValue>
            <KeyMaterial type="ByteString" value="00112233445566778899aabbccddeeff"/>
          </KeyValue>
          <CryptographicAlgorithm type="Enumeration" value="AES"/>
          <CryptographicLength type="Integer" value="128"/>
        </KeyBlock>
      </SymmetricKey>
    </ResponsePayload>
  </BatchItem>
</ResponseMessage>
```

Appendix D

XACML examples

D.1 Server-side policy example.

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy
PolicyId="urn:oasis:names:tc:xacml:2.0:policy5"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
<Target/>
<Rule
RuleId="urn:oasis:names:tc:xacml:2.0:rule4"
Effect="Permit">
<Target>
<Subjects>
<Subject>
    <SubjectMatch
      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
    <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#string">CloudClient</AttributeValue>
    <SubjectAttributeDesignator
          SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
          AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </SubjectMatch>
</Subject>
</Subjects>
<Resources>
<AnyResource/>
</Resources>
<Actions>
<Action>
    <ActionMatch
      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
    <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#string">Upload</AttributeValue>
    <ActionAttributeDesignator
          AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ActionMatch>
</Action>
</Actions>
</Target>
</Rule>
<Rule
RuleId="urn:oasis:names:tc:xacml:2.0:rule6"
Effect="Permit">
<Target>
<Subjects>
```

```
<Subject>
    <SubjectMatch
      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
    <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#string">CloudClient</AttributeValue>
    <SubjectAttributeDesignator
          SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
          AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </SubjectMatch>
</Subject>
</Subjects>
<Resources>
<AnyResource/>
</Resources>
<Actions>
<Action>
    <ActionMatch
      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
    <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#string">key-download</AttributeValue>
    <ActionAttributeDesignator
          AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ActionMatch>
</Action>
</Actions>
</Target>
</Rule>
</Policy>
```

D.2 Data-centric policy example.

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy
PolicyId="urn:oasis:names:tc:xacml:2.0:policy5"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">

<Target/>
<Rule
RuleId="urn:oasis:names:tc:xacml:2.0:rule5"
Effect="Permit">
<Target>
<Subjects>
<Subject>
    <SubjectMatch
      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
    <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#string">User-Hospital-A</AttributeValue
              >
    <SubjectAttributeDesignator
          SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
          AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </SubjectMatch>
</Subject>
</Subjects>
<Resources>
```

```
    <AnyResource/>
</Resources>
<Actions>
<Action>
    <ActionMatch
       MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
    <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string">sharewith-Hospital-D</
                AttributeValue>
    <ActionAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
            DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ActionMatch>
</Action>
</Actions>
</Target>
</Rule>
</Policy>
```

Bibliography

[1] National Institute of Standards and Technology. http://www.nist.gov/index.html. 38

[2] Storage Networking Industry Association, December 1997. http://www.snia.org/. 37

[3] Amazon. Amazon EC2 Instances. Online, October 2013. http://aws.amazon.com/ec2/instance-types/. 43

[4] Inc. Amazon.com. Amazon Elastic Compute Cloud, August 2006. http://aws.amazon.com/ec2/. 35

[5] S. Berger, R. Cáceres, K. Goldman, D. Pendarakis, R. Perez, J. R. Rao, E. Rom, R. Sailer, W. Schildhauer, D. Srinivasan, S. Tal, and E. Valdez. Security for the cloud infrastructure: trusted virtual data center implementation. *IBM J. Res. Dev.*, 53(4):560–571, July 2009. 14

[6] Richard Chow, Markus Jakobsson, Ryusuke Masuoka, Jesus Molina, Yuan Niu, Elaine Shi, and Zhexuan Song. Authentication in the clouds: a framework and its application to mobile users. In *Proceedings of the 2010 ACM workshop on Cloud computing security workshop*, CCSW '10, pages 1–6, New York, NY, USA, 2010. ACM. 27

[7] Martin Grooss Olsen Chris Piechotta, Adam Eno Jensen. Secure Dynamic Cloud-based Collaboration with Hierarchical Access. page 140, May 2012. 8, 11, 12

[8] ITI Technical Committee. *Document Encryption*. IHE International, Inc., August 2011. http://www.ihe.net/Technical_Framework/upload/IHE_ITI_Suppl_DEN_Rev1-1_TI_2011-08-19.pdf. 38

[9] Oracle Corporation. GlassFish Application Server, June 2013. https://glassfish.java.net/. 37

[10] CPUID. CPU-Z v.1.66. Online, August 2013. http://www.cpuid.com/softwares/cpu-z.html. 43

[11] H.A. Dinesha and V.K. Agrawal. Multi-level authentication technique for accessing cloud services. In *Computing, Communication and Applications (ICCCA), 2012 International Conference on*, pages 1–4, 2012. 27

[12] Um e Ghazia, R. Masood, and M.A. Shibli. Comparative analysis of access control systems on cloud. In *Software Engineering, Artificial Intelligence, Networking and Parallel Distributed Computing (SNPD), 2012 13th ACIS International Conference on*, pages 41–46, 2012. 15, 16

[13] William Burr William Polk Elaine Barker, William Barker and Miles Smid. *NIST Special Publication 800-57*. NIST, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899-8930, July 2012. `http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf`. 22

[14] Apache Software Foundations. Http components. Online, September 2012. `http://hc.apache.org/httpcomponents-client-4.2.x/index.html`. 39

[15] Damien Giry. Nist recommendations. Online, June 2013. `http://www.keylength.com/en/4/#Biblio10`. 38

[16] James Gosling and Sun Microsystems. *Java Programming Language*. Oracle Corporation, 1995. `http://www.java.com/`. 35

[17] S. Gupta, S.R. Satapathy, P. Mehta, and A. Tripathy. A secure and searchable data storage in cloud computing. In *Advance Computing Conference (IACC), 2013 IEEE 3rd International*, pages 106–109, 2013. 13, 14

[18] M. Hamdi. Security of cloud computing, storage, and networking. In *Collaboration Technologies and Systems (CTS), 2012 International Conference on*, pages 1–5, May. 8

[19] S. D. Hennessy, G. D. Lauer, N. Zunic, B. Gerber, and A.C. Nelson. Data-centric security: Integrating data privacy and data security. *IBM Journal of Research and Development*, 53(2):2:1–2:12, 2009. 17

[20] Hu, D. Ferraiolo, Kuhn, Information Technology Laboratory National Institute Standards, and Technology. Assessment of Access Control Systems, Interagency Report 7316. Technical report, National Institute of Standards and Technology, 2006. 15

[21] Luokai Hu, Shi Ying, Xiangyang Jia, and Kai Zhao. Towards an approach of semantic access control for cloud computing. In *Proceedings of the 1st International Conference on Cloud Computing*, CloudCom '09, pages 145–156, Berlin, Heidelberg, 2009. Springer-Verlag. 16

[22] Nguyen Thanh Hung, Do Hoang Giang, Ng Wee Keong, and Huafei Zhu. Cloud-enabled data sharing model. In *Intelligence and Security Informatics (ISI), 2012 IEEE International Conference on*, pages 1–6, 2012. 13

[23] I. Iankoulova and M. Daneva. Cloud computing security requirements: A systematic review. In *Research Challenges in Information Science (RCIS), 2012 Sixth International Conference on*, pages 1–7, May. 7, 8

[24] Michaela Iorga. Challenging Security Requirements for US Government Cloud Computing Adoption. Technical report, NIST, 2007. 8, 9

[25] M.R. Islam and M. Habiba. Agent based framework for providing security to data storage in cloud. In *Computer and Information Technology (ICCIT), 2012 15th International Conference on*, pages 446–451, 2012. 13

[26] Lishan Kang and Xuejie Zhang. Identity-based authentication in cloud storage sharing. In *Multimedia Information Networking and Security (MINES), 2010 International Conference on*, pages 851–855, 2010. 27

[27] G. Karjoth, A. Schade, and E. Van Herreweghen. Implementing acl-based policies in xacml. In *Computer Security Applications Conference, 2008. ACSAC 2008. Annual*, pages 183–192, 2008. 16

[28] A. Kumar, Byung Gook Lee, HoonJae Lee, and A. Kumari. Secure storage and access of data in cloud computing. In *ICT Convergence (ICTC), 2012 International Conference on*, pages 336–339, 2012. 13

[29] A. Kumbhare, Y. Simmhan, and V. Prasanna. Cryptonite: A secure and performant data repository on public clouds. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 510–517, 2012. 13, 14

[30] Bo Lang, Ian Foster, Frank Siebenlist, Rachana Ananthakrishnan, and Tim Freeman. A Flexible Attribute Based Access Control Method for Grid Computing. *Journal of Grid Computing*, 7(2):169–180, 2009. 20

[31] Jin Li, Gansen Zhao, Xiaofeng Chen, Dongqing Xie, Chunming Rong, Wenjun Li, Lianzhang Tang, and Yong Tang. Fine-grained data access control systems with user accountability in cloud computing. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 89–96, 2010. 14

[32] Dan Lin and Anna Squicciarini. Data protection models for service provisioning in the cloud. In *Proceedings of the 15th ACM symposium on Access control models and technologies*, SACMAT '10, pages 183–192, New York, NY, USA, 2010. ACM. 7, 12

[33] Hans Löhr, Ahmad-Reza Sadeghi, and Marcel Winandy. Securing the e-health cloud. In *Proceedings of the 1st ACM International Health Informatics Symposium*, IHI '10, pages 220–229, New York, NY, USA, 2010. ACM. 12

[34] Markus Lorch, Seth Proctor, Rebekah Lepro, Dennis Kafura, and Sumit Shah. First experiences using xacml for access control in distributed systems. In *Proceedings of the 2003 ACM workshop on XML security*, XMLSEC '03, pages 25–37, New York, NY, USA, 2003. ACM. 16

[35] Sun Microsystems. Sun's XACML Implementation, June 2006. `http://sunxacml.sourceforge.net/`. 36

[36] H.A.J. Narayanan and M.H. Giine. Ensuring access control in cloud provisioned healthcare systems. In *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, pages 247–251, 2011. 14

[37] H.A.J. Narayanan and M.H. Giine. Ensuring access control in cloud provisioned healthcare systems. In *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, pages 247–251, Jan. 10

[38] Tommi Henrik Nyrönen, Jarno Laitinen, Olli Tourunen, Danny Sternkopf, Risto Laurikainen, Per Öster, Pekka T. Lehtovuori, Timo A. Miettinen, Tomi Simonen, Teemu Perheentupa, Imre Västrik, Olli Kallioniemi, Andrew Lyall, and Janet Thornton. Delivering ict infrastructure for biomedical research. In *Proceedings of the WICSA/ECSA 2012 Companion Volume*, WICSA/ECSA '12, pages 37–44, New York, NY, USA, 2012. ACM. 8

[39] Timothy Grance Peter Mell. *NIST Special Publication 800-145*. NIST, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899-8930, September 2011. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf. 4

[40] T. Piliouras, Pui Lam Yu, Yang Su, V.K.A. Siddaramaiah, N. Sultana, E. Meyer, and R. Harrington. Trust in a cloud-based healthcare environment. In *Emerging Technologies for a Smarter World (CEWIT), 2011 8th International Conference Expo on*, pages 1–6, Nov. 7

[41] Raluca Ada Popa, Jacob R. Lorch, David Molnar, Helen J. Wang, and Li Zhuang. Enabling security in cloud storage slas with cloudproof. In *Proceedings of the 2011 USENIX conference on USENIX annual technical conference*, USENIXATC'11, pages 31–31, Berkeley, CA, USA, 2011. USENIX Association. 13

[42] Li qin Tian, Chuang Lin, and Yang Ni. Evaluation of user behavior trust in cloud computing. In *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, volume 7, pages V7–567–V7–572, 2010. 25

[43] S. Sankuratripati R. Griffin and R. Haas. Key Management Interoperability Protocol Specification Version 1.0., October 2010. http://docs.oasis-open.org/kmip/spec/v1.0/os/kmip-spec-1.0-os.pdf. 21, 28

[44] A.G. Revar and M.D. Bhavsar. Securing user authentication using single sign-on in cloud computing. In *Engineering (NUiCONE), 2011 Nirma University International Conference on*, pages 1–4, 2011. 27

[45] Arnon Rosenthal, Peter Mork, Maya Hao Li, Jean Stanford, David Koester, and Patti Reynolds. Methodological review: Cloud computing: A new business paradigm for biomedical information sharing. *J. of Biomedical Informatics*, 43(2):342–353, April 2010. 8, 9, 12

[46] R. Glenn S. Frankel and S. Kelly. The AES-CBC Cipher Algorithm and Its Use with IPsec , September 2003. http://tools.ietf.org/html/rfc3602. 38

[47] J. Saltz, T. Kurc, S. Hastings, S. Langella, S. Oster, D. Ervin, A. Sharma, T. Pan, M. Gurcan, J. Permar, R. Ferreira, P. Payne, U. Catalyurek, E. Caserta, G. Leone, M.C. Ostrowski, R. Madduri, I. Foster, S. Madhavan, K.H. Buetow, K. Shanbhag, and E. Siegel. e-science, cagrid, and translational biomedical research. *Computer*, 41(11):58–66, 2008. 60

[48] Pierangela Samarati and Sabrina De Capitani di Vimercati. Access control: Policies, models, and mechanisms. In *Revised versions of lectures given during the IFIP WG 1.7 International School on Foundations of Security Analysis and Design on Foundations of Security Analysis and Design: Tutorial Lectures*, FOSAD '00, pages 137–196, London, UK, UK, 2001. Springer-Verlag. 9

[49] Amit Sangroya, Saurabh Kumar, Jaideep Dhok, and Vasudeva Varma. Towards analyzing data security risks in cloud computing environments. In SushilK. Prasad, HarrickM. Vin, Sartaj Sahni, MahadeoP. Jaiswal, and Bundit Thipakorn, editors, *Information Systems, Technology and Management*, volume 54 of *Communications in Computer and Information Science*, pages 255–265. Springer Berlin Heidelberg, 2010. 14

[50] Jonathan Sharp. An application architecture to facilitate multi-site clinical trial collaboration in the cloud. In *Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing*, SECLOUD '11, pages 64–68, New York, NY, USA, 2011. ACM. 7

[51] A. Sirisha and G.G. Kumari. Api access control in cloud using the role based access control model. In *Trendz in Information Sciences Computing (TISC), 2010*, pages 135–137, 2010. 14

[52] M. Smith, M. Harbach, S. Mertins, A. Lewis, and L. Griffiths. Towards a translational medical research ecosystem. In *Digital Ecosystems and Technologies Conference (DEST), 2011 Proceedings of the 5th IEEE International Conference on*, pages 120–126, 2011. 60

[53] M. Smith, M. Harbach, S. Mertins, A. Lewis, and L. Griffiths. Towards a translational medical research ecosystem. In *Digital Ecosystems and Technologies Conference (DEST), 2011 Proceedings of the 5th IEEE International Conference on*, pages 120–126, 31 2011-June 3. 11

[54] Sebastian Speiser and Andreas Harth. Data-centric privacy policies for smart grids, 2012. 17

[55] Zhu Tianyi, Liu Weidong, and Song Jiaxing. An efficient role based access control system for cloud computing. In *Computer and Information Technology (CIT), 2011 IEEE 11th International Conference on*, pages 97–102, 2011. 14

[56] Totusoft. LAN Speed Test v.3.4. Online, December 2012. `http://www.totusoft.com/downloads.html/`. 43

[57] S. Trabelsi and J. Sendor. Sticky policies for data control in the cloud. In *Privacy, Security and Trust (PST), 2012 Tenth Annual International Conference on*, pages 75–80, 2012. 17

[58] Zhiguo Wan, Jun'e Liu, and R.-H. Deng. Hasbe: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing. *Information Forensics and Security, IEEE Transactions on*, 7(2):743–754, 2012. 14

[59] Changji Wang, Xuan Liu, and Wentao Li. Implementing a personal health record cloud platform using ciphertext-policy attribute-based encryption. In *Intelligent Networking and Collaborative Systems (INCoS), 2012 4th International Conference on*, pages 8–14, Sept. 7

[60] Guojun Wang, Qin Liu, and Jie Wu. Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 735–737, New York, NY, USA, 2010. ACM. 14

[61] Guojun Wang, Qin Liu, and Jie Wu. Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 735–737, New York, NY, USA, 2010. ACM. 15

[62] Kan Yang and Xiaohua Jia. Attributed-based access control for multi-authority systems in cloud storage. In *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*, pages 536–545, 2012. 14

[63] Yanjiang Yang and Youcheng Zhang. A generic scheme for secure data sharing in cloud. In *Parallel Processing Workshops (ICPPW), 2011 40th International Conference on*, pages 145–153, 2011. 13

[64] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *Proceedings of the 29th conference on Information communications*, INFOCOM'10, pages 534–542, Piscataway, NJ, USA, 2010. IEEE Press. 15

[65] Rui Zhang and Ling Liu. Security models and requirements for healthcare application clouds. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 268–275, July. 7

[66] Xiao Zhang, Hong tao Du, Jian quan Chen, Yi Lin, and Lei jie Zeng. Ensure data security in cloud storage. In *Network Computing and Information Security (NCIS), 2011 International Conference on*, volume 1, pages 284–287, 2011. 13

[67] Minqi Zhou, Rong Zhang, Wei Xie, Weining Qian, and Aoying Zhou. Security and privacy in cloud computing: A survey. In *Semantics Knowledge and Grid (SKG), 2010 Sixth International Conference on*, pages 105–112, 2010. 1