A Policy-Based Management Approach to Security in Cloud Systems

Nasser Abwnawar

School of Computer Science and Informatics

Faculty of Technology

De Monfort University

Leicester, UK

This thesis is submitted in partial fulfilment of the requirements for the

Doctor of Philosophy

February 2020

To my beloved parents

The thesis is dedicated to my loving father, Mr. Abdarrahman Abwnawar, who has been a great source of motivation, inspiration and endless support throughout my life, and who sacrificed a lot for me to be what I am now. It is also dedicated to my loving mother, Mrs. Manoba Ben Yzid who gave her love and support, for everything she sacrificed in her life for me. Without her loving care, prayers and support, it would have been very difficult for me to achieve my objectives.

To my beloved family

I would like to dedicate this thesis to my beloved wife Mrs. Magada Abwnawar and to my children, sisters and brothers. I owe everything I have achieved or will achieve to them. I hope that by obtaining my PhD I can put smiles on their faces.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Nasser Abwnawar
February 2020

Acknowledgements

In the name of Allah, the Most Merciful and the Most Gracious, I give praise and thanks to Him for supporting me with the strength to complete this research. This thesis could not have been completed without the recommendations, advice and suggestions of many people. It may not be possible to mention all of them here, but their contributions, guidance and support are extremely appreciated.

I should like first to thank my God Allah, without whom I could not even have contemplated all that was involved in this course of study.

My heartfelt thanks to my supervisors Prof. Helge Janicke, Dr. Richard Smith and Dr. Aboubaker Lasebae for their help and support during the period of my study. Their openness to new ideas, their extensive knowledge and their willingness to help me at every turn made them the best advisors and guide I could hope for.

And where would I be without my parents? Your loving care and nurture from the day I was born, as well as your teaching, are responsible for bringing me to this point, as well as your moral support and duaa during study of my PhD, for which I will be forever grateful.

Heartfelt thanks to my wife Magada, for the moral support and duaa. Heartfelt thanks to my son Ashraf and my daughter Maria for the moral support and duaa. Heartfelt thanks to my sons and daughters for the moral support and duaa. Heartfelt thanks to my sisters, my nieces and my brothers for the support and duaa. My heartfelt thanks to my best friends Ashraf, Hashem and Dr. Suwan for the moral support and duaa. My heartfelt thanks to my father-in-law and my mother in-law for the duaa. Thank you to all the staff in the STRL for

your support, and especially to Dr. Antonio Cau and Dr. Francois Siewe for support and advice during my thesis. My thanks to all of you. To all my friends, my deepest thanks for your help and support, and especially for your encouragement during my study.

Abstract

In the era of service-oriented computing, ICT systems exponentially grow in their size and complexity, becoming more and more dynamic and distributed, often spanning across different geographical locations, as well as multiple ownerships and administrative domains. At the same time, complex software systems are serving an increasing number of users accessing digital resources from various locations. In these circumstances, enabling efficient and reliable access control is becoming an inherently challenging task. A representative example here is a hybrid cloud environment, where various parts of a distributed software system may be deployed locally, within a private data centre, or on a remote public cloud. Accordingly, valuable business information is expected to be transferred across these different locations, and yet to be protected from unauthorised/malicious access at all times.

Even though existing access control approaches seem to provide a sufficient level of protection, they are often implemented in a rather coarse-grained and inflexible manner, such that access control policies are evaluated without taking into consideration the current locations of requested resources and requesting users. This results in a situation, when in a relatively ‘safe’ environment (e.g., a private enterprise network) unnecessarily complex and resource-consuming access control policies are put in place, and vice versa – in external, potentially ‘hostile’ network locations access control enforcement is not sufficient. In these circumstances, it becomes desirable for an access control mechanism to distinguish between various network locations so as to enable differentiated, fine-grained, and flexible approach to defining and enforcing access control policies for heterogeneous environments. For example, in its simplest form, more stringent and protective policies need to be in place as long as remote locations are concerned, whereas some constraints may be released as soon as data is moved back to a local secure network.

Accordingly, this PhD research efforts aims to address the following research question – *How to enable heterogeneous computing systems, spanning across multiple physical and logical network locations, as well as different administrative domains and ownerships, with support for location-aware access control policy enforcement, and implement a differentiated fine-grained access control depending on the current location of users and requested resources*?

To address this question, the presented thesis introduces the notions of ‘location’ and ‘location-awareness’ that underpin the design and implementation of a novel access control framework, which applies and enforces different access control policies, depending on the current (physical and logical) network locations of policy subjects and objects. To achieve, this the approach takes the existing access control policy language SANTA, which is based on the Interval Temporal Logic, and combines it with the Topological Logic, thereby creating a holistic solution covering both the temporal and the spatial dimensions. As demonstrated by a hypothetical case study, based on a distributed cloud-based file sharing and storage system, the proposed approach has the potential to address the outlined research challenges and advance the state of the art in the field of access control in distributed heterogeneous ICT environments.

List of Publications

1. Abwnawar, N., Janicke, H. Smith, R., A. Lasebae and Suwan, A. “Access Control in Cloud Environments: a Survey” In: DMU Doctoral Student Conference, May 2016, Leicester, UK.

2. Suwan, A., Siewe, F. and Abwnawar, N. “Towards Monitoring Security Aspects in Mobile Grid Computing Systems: a Survey” In: DMU Doctoral Student Conference, May 2016, Leicester, UK.

3. Suwan, A., Siewe, F. and Abwnawar, N. “Towards Monitoring Security Policies in Grid Computing: a Survey” In: IEEE Technically Sponsored SAI Computing Conference, July 2016, London, UK.

4. Abwnawar, N., Janicke, H. Smith, R. and A. Lasebae “Towards data privacy in heterogeneous cloud environments: an extension to the SANTA policy language” In: 2nd IEEE International Conference on Fog and Edge Mobile Computing (FMEC 2017), May 2017, Valencia, Spain.

5. Abwnawar, N., Janicke, H. Smith, R “Towards Location-Aware Access Control and Data Privacy in Inter-Cloud Communications” In: 17th IEEE International Conference on Smart Technologies (EUROCON 2017), July 2017, Ohrid, Macedonia.

List of Abbreviations

API	Application Program Interfaces
API	Application Programming Interface
ASL	Authorization Specification Language
AWS	Amazon Web Services
CPU	Central Processing Unit
DAC	Discretionary Access Control
EPS	Electronic Paper Submission
FAF	Flexible Authorisation Framework
FISMA	Federal Information Security Management Act
GPS	Global Positioning System
HTTP	Hyper Text Transfer Protocol
IaaS	Infrastructure as a Service
IBM	International Business Machines
ICT	Information and Communications Technology
IDE	Integrated Development Environment
IoS	Internet of Services
IoT	Internet of Things
IP	Internet Protocol
IT	Information Technology

ITL	Interval Temporal Logic
LAN	Local Area Network
LBAC	Lattice Based Access Control
MAC	Mandatory Access Control
NIST	National Institute of Standards and Technology
OS	Operating System
OWL	Web Ontology Language
PaaS	Platform as a Service
PAP	Policy Administration Point
PC	Personal Computer
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PHP	Hypertext Preprocessor
PIP	Policy information Point
PRP	Policy Retrieval Point
QoS	Quality of Service
RBAC	Role Based Access Control
REST	Representational State Transfer
SaaS	Software as a Service
SACPL	Semantic Access Control Policy Language
SCEL	Service Component Ensemble Language
SLA	Service Level Agreements
SOA	Service Oriented Architecture
SOC	Service Oriented Computing
SWRL	Semantic Web Rule Language
TCSEC	Trusted Computer System Evaluation Criteria

VPN	Virtual Private Network
WSDL	Web Service Description Language
XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Objectives:

- To briefly introduce the motivation behind the presented research.
- To briefly introduce the proposed approach.
- To formulate the research question, hypothesis, and thesis objectives.
- To provide an outline of the whole document.

1.1 Introduction

The multi-faceted issue of computer security has been seen as one of the top challenges both for the academia and the industry for several decades [18]. Since the 1970s, when information systems started exponentially growing in size and complexity, software became multi-tenant – that is, started simultaneously serving multiple users, which in turn led to an increased awareness of security-related issues [81]. Computer security is acknowledged as a highly complex and multi-faceted research discipline. Each of its sub-fields (e.g., user authorisation, user authentication, auditing, administration, data privacy, etc.) can be seen as an independent and challenging research area in its own right. Nevertheless, these sub-disciplines are tightly connected with each other, and are expected to be considered in a cross-cutting and interdependent manner. Security practitioners often metaphorically compare computer security to a chain – that is, similar to a chain that is as strong as its weakest link, computer system is as secure as its least secure component. In other words, a truly holistic solution addressing the security challenge is expected to consider all possible dimensions of this pressing problem in a cross-cutting manner.

1.2 Access control in distributed environments

Accordingly, one of such cross-cutting facets of the ICT security is access control. As outlined by computer security pioneers Sandhu and Samarati [82], "access control constrains what a user can do directly, as well as what programs executing on behalf of the users are allowed to do. In this way access control seeks to prevent activity that could lead to breach of security." From this perspective, access control is tightly coupled with user authentication. The latter is responsible for correctly identifying a user, whereas access control is typically based on the assumption that the user has been already identified, and, therefore, its main responsibility is to decide whether this specific user is actually allowed to access a particular resource. Thus, the effectiveness of the access control mechanism relies on and depends on the effectiveness of the underlying authentication mechanism [82]. In other words, if the user identity and the authentication mechanism are compromised, the access control component in most cases will consequently fail to prevent the malicious intrusion. In the same way, an access control mechanism is also supposed to be accompanied by a corresponding auditing/accounting mechanism. Broadly speaking, an auditing component is responsible for analysing user access requests and activities that took place within the managed computer system, and is typically implemented by logging relevant events for later analysis. These tight inter-dependencies between access control, user authentication and auditing components one more time illustrate the fundamental principle of designing and implementing ICT security solutions in a systematic and holistic manner.

There are two factors that contribute to the challenge of insufficient access control in modern enterprise ICT systems – namely, *i)* an increased adoption of cloud architectures, often spanning across multiple public and private clouds, thereby creating heterogeneous cloud environments, and *ii)* an ever-growing trend for hiring remote and/or mobile employers who often work from home using their own personal devices to access enterprise resources. These two factor are explained in more details below.

1.2.1 Impact of hybrid systems on security

As ICT systems exponentially grow in their size and complexity, fulfilling the access control requirements becomes an increasingly challenging task. As more and more users are able to simultaneously access shared resources from multiple locations, it becomes particularly important to protect sensitive information from unauthorised access – on the one hand, and maintain a high quality of service (QoS) and satisfy service level agreements (SLAs) by provide smooth and uninterrupted system operations – on the other hand. Admittedly, a representative example of such distributed multi-tenant computer systems are clouds.

Underpinned by advancements in networking and virtualisation technologies, Cloud Computing is widely recognised as one of the most critical Information Technology (IT) domains, where security and data privacy concerns play a crucial role in decision-taking processes [25, 43, 57, 87]. With cloud computing, traditional computer security issues are taken to the next level, underpinned by the increased system and networking complexity, multiplied by the extensive use of virtualisation technology. As enterprises' IT systems often span across multiple clouds and administrative domains, existing security mechanisms and models seem to be no longer suitable for cloud-hosted software and information. Open, virtualised and multi-tenant in their nature, the cloud computing paradigm has raised a number of formerly unknown research challenges, which are expected to be addressed by the academic researchers and industrial practitioners in the nearest future, to enable even more ubiquitous and pervasive usage of cloud computing. Accordingly, the key aspects that render existing computer security mechanisms less applicable to cloud-based scenarios are the following [25]:

- Cloud computing extensively uses the virtualisation technology to enable multiple tenants to simultaneously share one and the same physical space. This leads to new security breaches, associated with unauthorised access to private resources and information.

- The service-oriented nature of cloud computing assumes that different levels of a cloud solution stack may belong to different service providers. Such situations may potentially lead to a conflict of interests between various stakeholders, as there is no unified security framework, which all the interested parties could agree upon.

- Dynamic scalability and elasticity, virtualisation and service abstraction, and geo-physical location transparency – key characteristics of a cloud offering – result in a situation, in which hosted software systems have no fixed underlying infrastructure (due to frequent virtual machine migrations) and, therefore, security boundaries. In this light, it becomes a challenging task to identify and isolate a specific physical resource, which has been recognised as the one putting the system security at risk.

- In the age of Big Data, extreme amounts of raw data are being generated, processed, and stored in virtual clouds, meaning that corresponding security mechanisms have to be capable enough to cope with these amounts of data (which needs to be monitored and analysed for security purposes) and to maintain system security and a stable operational level.

These challenges become even more pressing, when more than one cloud provider is involved in the overall cloud-based software system. Multi-cloud deployments are known to provide more reliable, robust, and efficient ways of building complex software [8, 71]. However, along with these promising opportunities, come emerging challenges so as to how to maintain an appropriate level of security and data privacy, given the fact that considerable amounts of potentially sensitive business data is sent over potentially insecure public networks. Heterogeneous cloud environments typically do not allow implementing a single authorisation mechanism, a single policy language or a single enforcement mechanisms for customers, using several cloud providers or switching between private and public clouds. Each cloud provider typically has its own access control mechanism, which is limited in its flexibility to support other solutions.

For example, according to McAfee [48], enterprises using a multi-cloud strategy (typically composed of Amazon, Google and Microsoft clouds) have 14 misconfigurations on average. Some of these are related to multi-factor authentication not being enabled on one of the participating cloud platforms, Amazon S3 bucket encryption turned off, unused security groups, and Amazon Virtual Private Cloud flow logs being disabled. These cloud misconfigurations amount to about 2269 reported incidents of attempted illegal access per month for each enterprise.

1.2.2 Location impact on security

As more and more smart devices are used to access corporate services, it is important to consider where a device connects from, as it impacts upon the security. Within the corporate network there is an implicit level of trust, and access control mechanisms should allow access to certain data sources only from limited locations. If devices are transitioning from an untrusted to a trusted network then there needs to be additional controls in place to deal with this situation.

An important aspect contributing to these increased security and access control challenges is the increasingly popular practice of working from home and/or hiring remote employers, as well as the ‘bring your own device’ trend that allows employers use their own portable devices to work both from home and at the office. While all these trends deliver clear benefits for both a business and its staff, they also poses some challenges. The traditional tools and approaches designed to securely connect users with their applications and data stores are ineffective in the cloud and even, in many occasions, they become practically irrelevant.

This challenge can be illustrated by the following simple real-life scenario. Whether the applications and data are located in the company’s data centre or in the cloud, staff are likely to gain access by using a virtual private network (VPN). The problem is that VPNs were never designed to connect users to applications, but rather they were intended to connect networks to other networks. For this reason, bringing users from a remote network via the Internet into a trusted or secure private enterprise network so they can access an application or data is inefficient at best, and risky at worst. In fact, any enterprise offering its users access to its network through a VPN is significantly broadening its potential attack surface and elevating the risk of security issues. For example, a staff member whose device has been compromised can infect the network with malware that then quickly propagates through the private network as it scans for other resources and vulnerabilities to exploit. In such a situation, a mobile or remote employee is compromised externally and then connects back to the internal enterprise network, through which data-sensitive operations (e.g. banking or financial transactions) are executed. A company can significantly suffer from a security breach of this type, which will cause both financial and reputation losses. Furthermore, many companies tend to use a VPN to secure access to applications that are not even in their data centre, but reside on a cloud platform such as AWS or Azure. To achieve access, traditional

VPNs require extremely fine-grained traffic routing. This involves transferring data from the user to the corporate data centre, and then out via another VPN to the cloud provider before making its return trip back to the user. This results in a high response time and increased network latency, as well as an overall slow experience for the staff member, a challenge for IT administrators, and a technology that raises security concerns of its own.

As a result, the traditional VPN-based solutions for mobile and remote employers turn out to be ineffective and insecure, and require users to take different tedious actions depending on where they currently are and what applications they are accessing. For example, when a way from the office a user must connect to a VPN before accessing usual office applications such as a mailbox or a calendar, but when inside the office (i.e. accessing the same application from within the internal network) this is not required. The latter also does not guarantee smooth and seamless experience – for example, a user may use his/her personal tablet to access an application even from within the private network, but this attempt is denied because this device is not properly configured. Furthermore, there are also potential situations when the network that they are using simply does not support the communication requirements for a corporate VPN, or cannot deliver the bandwidth required for effective access without delays and interruptions.

Taken together, the increased adoption of hybrid multi-cloud architectures together with the increasingly popular trend of remote/mobile work introduce a previously-unseen level of security risks associated with unauthorised access to sensitive resources from various physical and network locations. In these circumstances, a desirable access control mechanism for an enterprise cloud system is expected to work with all data (i.e., formats, languages, representations) regardless of where data is located, while cloud users, both residing within and outside the private enterprise network, are expected to be able to control their access to their cloud-hosted data and resources. Indeed, given an increased amount of data transfers between private and public networks, there is an emerging and pressing concern of how sensitive business data is protected against unauthorised access across different computer systems, networks, physical locations, and administrative domains. In the presence of these multiple factors, it is important to propose a next-generation access control mechanism that would distinguish between various network locations to enable differentiated, fine-grained, flexible approach to defining and enforcing access control policies for heterogeneous environments. Arguably, in its simplest form, more stringent and protective policies need to be in place as long as remote locations are concerned, whereas some constraints may be released as soon as data is moved back to a local secure network. A more advanced solution also has to distinguish between trusted and untrusted devices, which are used by customers to access corporate resources from within the private network and from an external public network location (possibly through a VPN).

1.3 Research question, hypothesis, and objectives

Following the motivation of insufficient support for applying location-aware access control policies, which would differentiate between different network locations, we formulate the following research question to be addressed by the presented research effort:

"How to enable heterogeneous computing systems, spanning across multiple physical and logical locations, as well as different administrative domains and ownerships, with support for location-aware access control policy enforcement, and implement a differentiated fine-grained access control depending on the current location of users and requested resources"?

To address this research question, we are also putting forward the following hypothesis that is intended to be proved in the context of this research:

"The outlined challenges can be potentially addressed by extending the existing functionality of access control tools and languages with native support for detecting the current location of protected resources, as well as of users trying to access them, and thereby en-

abling these existing tools to apply and enforce differentiated fine-grained access control policies with respect to the current location”.

To provide a more structured and systematic overview of the presented research work, as well as to evaluate the final research results, it also makes sense to split the overall work into several theoretical, technical and experimental objectives, as described below.

1.3.1 Theoretical objectives

- To study the state of the art in the domain of cloud computing and heterogeneous ICT environments with a specific focus on existing challenges, specifically related to access control.
- To identify an existing research and technological gap to be addressed by the proposed research.
- Formulate the main challenges within the identified gap to be addressed by the proposed research.
- To propose a potential approach to address the identified gap and challenges.

1.3.2 Technical objectives

- To design and implement a software prototype of a location-aware access control mechanism for heterogeneous ICT environments.
- To design and implement a hypothetical case study to validate the viability of the proposed approach and the developed software prototype.

1.3.3 Experimental objectives

- To validate the viability of the proposed approach and the developed prototype through a case study with respect to several key criteria.
- To measure and evaluate the results of the validation. The outlined research question, hypothesis and objectives will be re-visited in the concluding chapter of this thesis. They will serve to evaluate the research results – that is, we will discuss whether (or to what extent) the presented PhD work has achieved the specified objectives, proved the hypothesis and answered the main research question.

1.4 Thesis contributions

Achieving the goals of the proposed approach primarily contributes to the research areas of computer security and access control – in general, and access control for heterogeneous/hybrid environments – in particular. The approach puts forward the novel concepts of access control object/subject location as well as location-awareness as a key characteristic of a policy enforcement mechanism responsible for evaluating policies and taking access control decisions. Neither of these features have been previously proposed, discussed or implemented, which makes the proposed approach a potentially valuable contribution to a wide range of academic researchers and industrial practitioners. Moreover, the contribution of the thesis also spans across several adjacent research fields, such as Cloud Computing (and especially – Hybrid Cloud Computing), as well as provides a new application domain for the existing logical formalisms, including the Interval Temporal Logic and the Topological Logic. More specifically, the main contributions of the described research effort can be summarised as follows:

I. Literature survey of the state of the art in access control in (hybrid) cloud computing – as part of fulfilling the theoretical objectives, a literature survey has been conducted, identifying existing limitations and gaps in the considered research field. As it became clear, the challenging topic of insufficient support for taking into consideration subject and object location when enforcing access control policies is yet to be explored. The conducted survey, as well as the whole presented research work in general, is intended to raise the overall awareness within the research community and attract further attention to this motivating and challenging problem.

II. Definition of functional requirements for a location-aware access control system – based on a thorough investigation of the existing access control systems, approaches, and techniques, some existing limitations have been identified. These limitations, in turn, led to devising a list of functional properties for an envisaged solution. Briefly, these include support for modelling and extraction location-related information, as well as an ability to enforce access control policies with respect to physical and logical locations of both subjects and objects. This functional specification serves as the key reference underpinning the design and implementation of the future system. Moreover, it also contributes to the state of the art in enabling location-aware access control policy enforcement, as it is expected to be re-used by the wider research community, willing to engineer their own solutions based on the proposed approach (i.e., and thus not 're-inventing the wheel').

III. Novel concepts of location, location-awareness and policy transition – these novel concepts have been proposed to address the requirement of enabling differentiated treatment of resources and users depending on their contexts. More specifically, it was important to introduce and clearly define the main concepts, so as to be able to further build the whole approach based on them. These concepts are seen as contributions, because previously there has been little evidence of integrating the spatial dimension into the context of access control policy enforcement. Discussing these concepts in this thesis will hopefully contribute to creating next-generation logically-underpinned access control systems in the future.

IV. Design and prototype implementation of the location-aware access control system – using the proposed system, one is expected to benefit from the possibility to enable location-aware access control policy enforcement. Moreover, the outlined functional specifications underpinned the conceptual design of the proposed system. In the future, it has the potential to act as a reference model for the wider research community, who are willing to implement their location-aware access control policy enforcement. As far as the prototype implementation is concerned, we have developed a prototype version of the proposed system, which serves to demonstrate the viability of the whole presented approach. Using this system, users are expected to benefit from the possibility to enable location-aware access control policies in heterogeneous computer environments. Moreover, since the current implementation follows an open-source approach to software development and distribution, users are also encouraged to further extend the existing functionality to implement required emerging features, and thereby act as contributors to our system.

1.5 Outline of the thesis

Chapter 2 serves to provide a brief overview of Cloud Computing. Starting from Service-Oriented Computing, it explains the existing models and definitions, as well as summarises advantages. It also looks into existing challenges, among which access control remains one of the most pressing issues.

Chapter 3 first provides the reader with theoretical foundations of Access Control in computer systems. It then proceeds with a literature survey of relevant research works in the field of access control for cloud environments. The chapter concludes with a discussion, identifying an existing research gap to be addressed by the present research.

Chapter 4 presents some further background information by briefing the reader on the theoretical underpinnings of the SANTA policy language – a baseline on top of which the proposed access control functionality will be built. The chapter also highlights the existing limitation of SANTA, i.e., lack of support for capturing the spatial dimension, and introduces Topological Logic as a way of addressing this limitation.

Chapter 5 presents the architecture of the proposed access control framework, enhanced with support for location-awareness and policy transition. Based on the existing XACML reference architecture, the chapter describes the conceptual design of the access control framework.

Chapter 6 goes into implementation details of the proposed Location-Aware Access Control framework and presents a sample case study to prove the proposed concepts and demonstrate the viability of the developed prototype framework. More specifically, the case study is based on a scenario, where enterprise resources are accessed from various, not necessarily trusted locations. Accordingly, the developed prototype is able to detect location and, based on policies in places, decide if the access should be granted.

Chapter 7 evaluates and discusses the results of the conducted case study. It also revisits the main research question, hypothesis and thesis objectives in order to evaluate the overall achievements of the presented research effort.

Chapter 8 summarises the whole documented by presenting some final remarks and outlining several directions for future work.

Chapter 2

Background Theory: Cloud Computing

Objectives:

- To review the evolution from Service-Oriented Computing to Cloud Computing.
- To review existing definitions, models and advantages of Cloud Computing.
- To review existing challenges of Cloud Computing.

2.1 Introduction

The aim of this chapter is to familiarise the reader with the notion of cloud computing – the main context of the presented research effort. The chapter first briefs the reader on the origins and history of cloud computing by describing its relation to Service-Oriented Computing (SOC) and grid computing. Then, the chapter proceeds with an actual overview of the cloud model; it defines the concept and presents classification taxonomy of clouds with respect to the delivery model and the deployment model. It also outlines a list of main characteristics of cloud computing, followed by a (non-exhaustive) list of potential benefits it can offer its customers. The last part of the chapter is dedicated to raising and discussing main challenges and gaps currently existing in the domain of cloud computing, among which data privacy and security is seen among the most pressing. By thoroughly looking at the issues of data privacy and security, we aim at explaining the motivation behind the presented research work and provide the reader with understanding of how important the existing problems are. This last part of the chapter is also acting as an introduction to the next chapter, in which we will survey the state of the art in relevant research field, identify existing gaps and limitations, and position our own work respectively.

2.2 From service-oriented computing to cloud computing

The areas of software development and delivery of IT services have experienced several paradigms shifts in the last three decades. After the introduction of Object-Oriented Computing in the 1980s [77] and component-based software engineering [26], modern IT systems are now being transformed into Service-Oriented Architectures (SOAs). The SOA employs an evolutionary approach to engineering complex and distributed software systems in a technology-agnostic and loosely-coupled fashion. Service-orientation inspires from and further develops the main principles underpinning the success and wide adoption of object-oriented and component-based software development. Among other things, these principles primarily include self-description, run-time functionality loading, and explicit encapsulation.

The SOA paradigm relies on using services as atomic building blocks. In a broad sense, a service in the SOA can be defined as a reusable software component, remotely accessible in a loosely-coupled and technology-agnostic way using standardised and well-defined interfaces and protocols [41]. To fulfil these requirements, services constituting a service-based application system are typically designed to perform granular individual functions with limited awareness of how other components (i.e., services) of the entire system are actually implemented, functioning and interact between themselves. From this perspective, the SOA can be considered as a design pattern for engineering distributed and complex computer systems, which relies on an efficient, yet simple principle – distinct reusable pieces of software functionality are provided via established and standardised channels (i.e., protocols and interfaces) over the network. Thanks to this principle, the main advantages of the SOA are becoming obvious – distributed software systems can be constructed from already existing, functioning and reliable pieces in a loosely-coupled manner so as to be easily modified when/if required. By abstracting the underlying implementation and only exposing their Application Program Interfaces (APIs), multiple heterogeneous services can be seamlessly assembled into a single service-based application system, which typically does not depend on the implementation specifics of individual services [32, 34]. In other words, to be widely re-usable services are expected to be technology-, product-, and vendor-agnostic.

It is important to bring to the reader's attention the fact that the notion of services implies that they are typically designed and implemented to remain relatively static and stable, whereas the configuration of a service-based composition – that is, the way services are interconnected and interact between themselves – is expected to be dynamic and can freely change and evolve with the time. In other words, situations, where emerging business processes introduce new requirements for the supporting software system, are not expected to trigger serious and potentially effort-intensive manual changes to the software source code.

In this context, services can be seen as 'black boxes', which expose their interfaces and descriptions so as to enable applications and users, which are not necessarily aware of the actual implementation details, to discover and access their provided functionality over the network. Services are rather stable and static components, relying on standardised, well-defined and technology-independent interfaces and self-description notations for discovery and remote access. A standardised and widely adopted format for describing Web services, for example, is Web Service Description Language (WSDL) – an XML-based and completely platform-agnostic language. The Representational State Transfer (REST) architecture is another way of implementing Web services. It is also completely technology independent, as it relies on the standard commands of the HTTP protocol (e.g., GET, POST, DELETE, etc.).

The described SOA principles, if correctly implemented, lead to a whole new paradigm in modern computing – namely, Service-Oriented Computing (SOC). In SOC, enterprise software is organised and implemented in such a manner that distinct pieces of the overall distributed service-based application system maybe managed and controlled by different ownership domains and companies [61]. One of the main advantages of organising IT systems in the service-oriented manner is the business agility – that is, business processes implemented as service-based and loosely-coupled software are typically expected to be modified and evolve in a much easier and effortless manner. As opposed to traditional 'monolithic' approaches to designing and implementing IT systems, SOC is not constrained with the underlying technologies, which typically requires much more time to be adjusted when/if required.

SOC and its promising business opportunities attracted even more attention in the last decade, as many companies, including large enterprises and small businesses, have been trying to handle rapidly changing requirements of the market, such as, for example, evolving customer needs or minimum up-front investments to implement new business processes [97]. The competitive market often requires from organisations to modify their IT systems by retiring outdated components and integrating new functionality in a continuous and rapid manner to minimise 'time to market'. As it soon became evident, SOC provided promising opportunities to address these requirements by enabling easily modifiable and loosely coupled business processes. SOC allowed companies to rapidly develop or modify their software systems each time an emerging business requirement arises by re-using existing building blocks – i.e., services – to build low-cost, yet reliable software [97].

The wide adoption of the service-oriented model for building IT systems from re-usable services, the recent advances in the mobile and networking technologies making the Internet more and more accessible and ubiquitous [13], and the development of Web 2.0 [83] – a trend in designing and implementing Web sites enhanced with rich support for social networking, eventually led to the emergence of the so-called Internet of Services (IoS) [21]. The IoS aims at creating a global connected infrastructure, where every IT resource is available as a discoverable and easily accessible Internet service. The notion of IT resources in this context is quite broad and goes beyond the traditional software; it also includes various development tools and the underlying hardware infrastructure required to deploy and run the software (i.e., servers, storage and network). Taken together, all these software and hardware resources, offered as services remotely to the end user, are nowadays known as cloud computing.

2.3 Cloud computing: definitions, models, and benefits

More and more organisations have been attracted by the described advantages of SOC and started engineering their IT systems based on the SOA principles. Thus, they were enabled to minimise human and time effort required to implement specific software components from scratch by re-using already existing, functioning, and optimised service solutions, accessible over the network. The next step in this evolution from the traditional in-premises way of running enterprise IT systems was the move to cloud computing, which offered its consumers even more possibilities to remotely access processing, storage and network resources. With its emergence and further development, conventional computing has finally evolved into a model, where IT resources are commoditised and delivered over the network in a manner, similar to traditional utilities, such as electricity and water [22].

The concept of ‘computing as a utility’, however, is not completely novel and was first introduced more than 50 years ago by John McCarthy [46, 39]. Admittedly, back then in the 1960s the technologies were not advanced enough to implement this visionary idea, and it took almost another 30 years to make first steps towards the actual implementation of the cloud computing concept [10]. The things first started changing rapidly in the 1980s with the mass production of affordable personal computers by IBM. Using computers at home for personal use also required using appropriate operating systems, which were released by Microsoft and made desktop PCs even more ubiquitous and widely used by ordinary users both at work and at home. The next step were the advances in networking technology, which enabled connecting individual computers into networks and eventually led to the creation of the Internet [64]. Supported by various interoperability standards and advances in software development, the Internet created a convenient environment for running on-line businesses and other types of commercial activities, such as offering remote IT services over the Internet. For example, Salesforce.com[1] in 1999 was the first company to deliver sales automation software to end users through a Web site on a pay-per-use basis. In 2002, Amazon launched its Amazon Web Services[2] (AWS) – a multi-layered platform for accessing Web-based storage and processing services. Similarly, during the next several years other major IT providers started including cloud solutions as part of their commercial offering. Finally, by 2009-2010 the cloud computing market had finally shaped with the main players getting on-board. Companies like Google, Microsoft, IBM, Oracle, Salesforce.com, etc. started offering a broad range of cloud solutions suitable for various needs of individual users, small organisations and large enterprises. The term ‘cloud computing’ was becoming more and more familiar not only to IT professionals and computer scientists, but to a much wider audience of ordinary users.

Despite its high popularity and rapid development, there seems to be a lack of a common agreed definition of cloud computing. To date, there can be found several definitions of the term in the literature, which seem to highlight various important aspects of this concept [10, 95]. For example, National Institute of Standards and Technology (NIST), US, proposed one of the most widely used and extensive definitions [62]:

[1] http://www.salesforce.com/

[2] https://aws.amazon.com

"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."

A slightly different definition is offered by Gartner [72], who define Cloud Computing as follows:

"A style of computing where massively scalable IT-enabled capabilities is delivered 'as a service' to external customers using Internet technologies."

Similarly, Cisco also proposed its own definition of clouds with an emphasis put on the aspects of virtualisation and multi-tenancy [11]:

"IT resources and services that are abstracted from the underlying infrastructure and provided 'on-demand' and 'at scale' in a multi-tenant environment."

2.3.1 Cloud service sharacteristics

All these definitions above, albeit different from each other, can be uniformly summarised with outlining the following five key characteristics of a cloud service [62]:

- *Resource pooling* refers to the internal organisation of the cloud environment, in which (virtualised) resources are 'pooled' together so as to be simultaneously provisioned to multiple consumers by means of the multi-tenant sharing model. Resource pooling enables dynamic assignment and re-assignment of physical and virtual resources to support continuously changing demands of the consumer. This important feature of cloud computing is supported by the recent advances in the virtualisation discipline, which enables abstraction from underlying physical resources. Also, resource pooling and virtualisation bring the concept of location-independence, which means that cloud consumers are typically unaware of the exact geographical location of where their applications and data run and are stored respectively – this is achieved by pooling together resources belonging to different physical data centres or servers.

- *Rapid elasticity* describes the ability of cloud services to be (automatically) reserved and released so as to support scaling in and out of the deployed software, whose resource demands may dynamically change. Admittedly, the overall amount of virtual resources offered by a cloud platform is always limited by the underlying physical resources. However, using virtualisation techniques to enable cloud elasticity, together with resource pooling and multi-tenancy, creates an impression of seemingly infinite computing utility from the consumer's perspective.

- *On-demand self-service* suggests that the cloud client expects that cloud-hosted software is served with sufficient cloud resources (e.g., computation, storage, network, etc.) whenever it is required – that is, on demand – in a completely automatic way, without human interaction.

- *Broad universal network access* refers to the capabilities of a cloud service to be accessed remotely over the network (i.e., the Internet) by means of standard interfaces and protocols. This feature also refers to the cross-platform access to the cloud – that is, from a wide range of client devices and platforms (e.g., desktop computers on Windows and Linux, tablets and mobile phones on Android, IoS and Windows, etc.)

- *Measured service* is the last but not the least key feature of cloud computing, which means that consumers are expected to pay only for the actual consumption the cloud service. Calculation and billing of the actual consumption has to be done in a transparent and fair manner. To achieve this, cloud platforms are typically equipped with automatic tools for consumption monitoring and reporting, which enable prompt metering and, eventually, billing.

2.3.2 Cloud deployment models

Traditionally, there are four different deployment models of cloud computing, identified in the literature [10]:

- The traditional understanding of cloud computing, where clients can access and use remotely located virtualised services, is usually referred to as the *public* deployment model. With this model, the cloud services are publicly available to virtually anyone (i.e., an individual user, a small company, a large-scale enterprise, a governmental organisation, etc.), who intends to acquire remote computational/storage resources via the Internet.[3]
- For large-scale multi-departmental enterprises, it is often more economically attractive to build up and manage their own cloud data centre within their premises. Accordingly, this deployment model has become known as the *private* cloud, meaning that cloud services are only expected to serve the needs of a specific organisation and are not typically available to the public. The rationale behind running a private cloud data centre is two-fold. First, if individual departments of a large enterprise (multi-national) company demand for elastic computational and storage resources, which are expected to be fully utilised, this can bring certain economic benefits when compared to expenses for third-party commercial cloud services (in these circumstances, however, thorough calculations are needed so as to make sure that private deployment is indeed a better option allowing to save money). Second, there are often cases when enterprises do not want to release sensitive business data beyond their private network perimeter, which will most probably happen with the public cloud deployment. Accordingly, in circumstances, when a move to a public cloud is hindered with privacy and security

[3] This is how the term 'cloud' appeared – a cloud-like shape was frequently used to denote the Internet in various network diagrams, where specific details of the internal organisation of the Internet were not particularly essential.

concerns, it might make more for an enterprise to run a local private cloud under its control, thus making sure that no sensitive data is released to the outer world.

- A *hybrid* cloud, where the IT system of an enterprise partially runs on public and private clouds, also exists. With this hybrid model, enterprises can find a right balance between their security and financial concerns. For example, it is possible to run the most critical and sensitive tasks in the local cloud (e.g., financial software), whereas less critical ones can be sent to the public cloud to save on expenses required to invest in a more powerful private data centre.
- Another deployment model is known as the *community* cloud, which represents the case, where a cloud infrastructure is set up to serve the requirements of a specific community or a group of organisations with shared interests and requirements (e.g., in terms of privacy, organisational policies, functionalities, standards, etc.). With this deployment model, a public cloud platform can only be accessed by a limited number of parties. Examples of such community clouds include various governmental clouds (i.e., cloud resources are only accessible by governmental and state organisations) or research clouds (i.e., cloud resources are only accessible by participating universities and research centres).

2.3.3 Cloud delivery models

When considering to move an IT system to the cloud, enterprises should clearly understand and distinguish between several cloud service models, which primarily determine the type of resources to be provisioned to the consumer. Historically, the following three service models are identified [62]:

- *Infrastructure as a Service* (IaaS) is usually seen as the bottom layer of cloud computing service model. It refers to enabling cloud consumers with on-demand access to low-level infrastructure resources, such as processing power, storage, network bandwidth, etc. These resources are typically offered to the consumer as a bare virtual machine running on top of a physical server. IaaS subscribers are then expected to install an appropriate OS, middleware to run their software, and the actual software to be executed in the cloud and accessed remotely over the network. Accordingly, IaaS users have access to and can control these deployed components (i.e., OS, middleware, deployed software and business data), whereas management of the physical hardware components is typically beyond their control. Typical consumers of an IaaS cloud offering are companies attracted by the promising opportunities to save on acquiring and maintaining potentially expensive hardware by using cloud-based hardware resources only when they are actually needed and without major up-front investments. Accordingly, they are charged and billed based on monitoring of low-level metrics such as the number of virtual machines, CPU and memory consumption, storage consumption, network bandwidth, etc.

In this setup, assignment of necessary network configurations, storage space and processing capacities is done in the cloud and these are provided as cloud services in a standard manner. The cloud resources such as storage space, equipment for networking and servers are provisioned from a pool of shared resources. Typically, users are expected to configure and manage applications on their own, rather than rely on the service provider in this setup. GoGrid and Amazon EC2 are some of the representative IaaS service providers.

- *Platform as a Service* (PaaS) is typically seen as the middle layer of cloud computing. This delivery model enables provisioning of a computational platform (i.e., an operating system equipped with pre-deployed middleware and a technological stack required to install and run arbitrary software), thus enabling its subscribers to avoid the cost and complexity of buying, configuring and managing underlying hardware infrastructure and the computational platform itself. These benefits make PaaS an attractive option for software vendors, enterprises' IT departments, and individual software developers – all these PaaS clients are willing to avoid the routine of managing the infrastructure and the platform (thus also saving on their expenses) and concentrate on their immediate software development tasks. In these circumstances, the PaaS provider is responsible for monitoring deployed applications and allocating infrastructure and platform resources as required by the software. To charge its customers, PaaS providers typically rely on collecting data, about CPU and memory consumption. With the emergence of add-on marketplaces, PaaS providers might also charge for using such add-on services as messaging queues or databases. These services provide not only application runtime environments, but also an IDE (Integrated Development Environment) to their users, giving them the flexibility to develop applications to meet their requirements and related configurations. Therefore, PaaS serves its customers with multiple application development services. In this arrangement of serving, the application development framework is provided as an abstract service so that users can build services based on that platform. Users have the capability to develop applications based on their requirements and run them in the environment provided by the service provider. A common set of OSs and software stacks, such as Ruby, LAMP (bundled Apache, MySQL, PHP for Linux), J2EE components, etc. comes bundled together in PaaS.

- *Software as a Service* (SaaS) is seen as the top level of cloud computing, which enables its customers to access and use cloud-based software, usually in a cross-platform (e.g., from PCs, laptops, smartphones, etc.) and on-demand fashion. This delivery model can be seen as a transition from the traditional and established way of delivering software available as a product (i.e., distributed physically in boxes), which requires installation, configuration and continuous maintenance from the client. On contrary, in SaaS a single software code base is ‘shared’ among multiple users, who can simultaneously access individual instances of that particular software remotely over the network. In SaaS, clients are almost completely exempted from the system management routine – they are only responsible for configuring their respective instances of the software and personal business data. All the management tasks concerning underlying resources, the platform and the software are typically beyond their control, and remain the responsibility of the SaaS provider. The applications based on cloud come as an integrated solution to reduce the time and money cost involved in the installation of a separate hardware systems, licensing and updating of software when providing various IT products and services for the users. SaaS serves the customers as needed, providing a comprehensive set of applications to work on. For a given application, many users can share only one instance running on the server. It is a ‘win-win’ situation for both consumers and service providers such as Salesforce.com, Google and Microsoft since the former need not to worry about the cost of installation and licensing, while the latter needs to have capabilities of running only one instance, serving many users in parallel. Table 2.1 presents a summary of these service models.

2.3.4 Cloud computing benefits

Let us now consider in more details the main benefits cloud computing may offer to its customers. These benefits of using cloud computing are manifold, and we can distil the following (non-exhaustive) list of main advantages associated with switching from the traditional in-premises model for accessing computational resources to a cloud-based solution [86]:

- *Resource elasticity*: elasticity is seen as one of the key advantages of cloud computing. It refers to the ability of a cloud platform to expand and contract automatically – that

Table 2.1 Cloud service delivery models.

Cloud Service	Functionality	Example Service Providers
SaaS	Applications, which can be extended as needed, are hosted in the cloud and made available for consumers through the Internet, as services.	Salesforce.com, Google Drive
PaaS	Consumers are provided with a comprehensive environment for designing, developing, testing and deploying applications in the cloud.	Google App Engine, Heroku, Amazon Elastic Beanstalk
IaaS	On-demand serving of file space, computing power and database management, etc. in a pay-as-you-go manner.	Amazon Elastic Compute Cloud (EC2), GoGrid

is, to reserve and release computational resources, such as CPU, memory or network resources – upon capacity demands. To support elasticity and ensure that appropriate service levels are maintained, cloud platforms continuously monitor the usage of deployed applications and available resources. In response to reaching a critical level of CPU/memory utilisation, additional computational instances can be launched and incoming user requests can be spread across instances evenly. An important point here is that there is no need to over-provision resources for the peaks. In some cases, the process of adding/removing computational instances is managed by the cloud platform in a completely automated way, whereas in other cases, application developers are expected to build such capability by themselves or integrate it into their applications' source code using appropriate cloud platform APIs. Resource elasticity is of particular importance to small business who cannot yet afford acquiring and maintaining a whole data centre, but whose IT demands are constantly growing and, therefore, have to be properly addressed. The cloud organisations could cushion cyclical or seasonal effects or fluctuations in IT demands during specific peak periods by provisioning scalable resources. Mainly for small business for which the in-house IT is running on tight budgets, cloud computing services could enhance the competitiveness in their business.

- *Cost saving*: a consequent advantage of provisioning computing resources only when they are really needed is cost saving. As opposed to the traditional way of running a data centre within a company, which requires major up-front investments into hardware and software, the cloud model allows for an immediate gain. In other words, by adopting the cloud computing model, companies move from a capital investment to an operational expense. It is worth noting that cost savings also include potential savings on technical personnel required for managing the IT data centre, utility bills and expenses for electricity and cooling, licensing, etc. Taken together, these factors are most likely to become the key factor for company decision takers when considering migration to clouds. There are many reasons for attributing low-cost cloud technology. The model of billing is pay-per-use; infrastructure is never bought and thus maintenance is lower. Recurring and initial expenses are considerably lower compared with traditional computing models.

- *Business agility*: in a situation, where companies are not constrained with the capacities of their in-house data centre, it is becoming easier to make changes to the IT system of organisation so as to meet emerging business requirements. It means that whenever a company realises that increasing computational demands have to be properly handled, it can simply request more cloud capacities, which will be elastically provisioned in a seamless, transparent and automatic manner. On contrary, consider a situation where the very same company would have to go through the tedious process of acquiring and configuring additional servers and equipment and integrating them into the existing data centre. Besides the agility at the infrastructure level (i.e., processing, storage, and network resources), companies can also benefit from agility at the platform and software levels – for example, they are enabled to switch between various programming languages, run-time environments, databases, etc. – all these changes are done rapidly and often simultaneously. All these factors allow organisations of any size to react to dynamic market changes and, therefore, fully utilise emerging business opportunities.

- *Increased reliability, built-in disaster recovery and back-ups*: with cloud computing, the task of managing technology is placed on the technology provider. It is their responsibility to provide such features as built-in data protection and replication, fault tolerance, self-healing and disaster recovery as part of the Service Level Agreement (SLA) with the customers. As a result, cloud consumers are exempted from these routine (yet costly and time-consuming) tasks and can concentrate on their immediate business goals.

- *Remote access from any location and any device*: one of the key advantages of cloud computing is that it is enabling greater opportunities for device independence, portability, interconnection and collaboration. With applications and data hosted in the cloud, as well as recent development of the Internet of things and ubiquitous connectivity, it becomes much easier to enable users to access systems regardless of their location or devices they are using [30]. With the growing popularity of smartphones, tablets and other hand-held devices, there is also an increasing need for data access on the go. Employers are no longer required to be physically present at the company office to do their job – their everyday tasks can be done remotely from home, in an airport, on the train, etc. With cloud computing, virtual offices can be quickly set up; employees can easily work from home; travelling salespeople can have all their data available in any location without being physically present at the office. As a result, companies can save costs on the office space and also hire remotely-located professionals in ‘less expensive’ regions of the world.

- *'Green' factor*: the last but not the least factor making companies migrate their IT systems to cloud environments is the ever-pressing ecological concern. In a cloud computing environment resources are shared between multiple customers, which results in a more optimised consumption of the available resources for a similar energy cost. Moreover, for multi-national corporations spanning across the globe and different time zones, the computing power staying idle at one geographic location during off-work hours (i.e., at night) could be seamlessly harnessed by the branches located on the other side of the globe, which are currently operating. This reduces not only the power consumption, but also the amount of physical hardware required, and, therefore, can reduce the overall footprint on the global ecology.

2.4 Cloud computing challenges

A holistic overview of cloud computing would not be completed without listing and discussing its potential shortcomings and disadvantages of migrating IT systems to cloud environments. Accordingly, these potential disadvantages include [86, 38, 99]:

- *Dependency on the network connection* concerns the inevitable requirement for companies to have an established and reliable broadband Internet connection. Despite the considerable advances in the networking and mobile technologies, this requirement is becoming less and less strict, unstable and low-speed network connections are seen among the key factors preventing the adoption of the cloud paradigm. Additionally, adopting the cloud-based model inevitable results in modifying the architecture of the enterprise software to minimise the dependency of the software performance on the network latency, which in turn leads to certain financial expenditures.

- *Service availability* is another concern for enterprises considering moving to the cloud. As reported by independent monitoring web sites,[4] cloud providers still cannot guarantee 100% up-time and un-interrupted performance of the hosted applications as part of the offered SLAs. In the situation when a one-minute down-time may result in critical losses both in terms of financial revenue and business reputation, companies may decide to run the own data centres in order not to be dependent on unreliable cloud provider and avoid such critical situations.

- *Complicated migration process* is another factor slowing down the adoption of the cloud model. Moving an existing software system to a cloud environment is not as easy and straight-forward as it might first appear. Cloud providers impose different standards and policies, which make existing application not immediately transferable to the cloud. In other words, cloud consumers are typically expected to make some changes to their software before deploying to the cloud. Moreover, in some cases migration of a legacy system might even turn out to be impossible due to underlying dependencies and architectures.

- *Data location and privacy* are the two critical concerns when it comes to sensitive information (e.g., governmental, business, health, etc.). The cloud model suggests that data has to be transferred and stored on physical servers somewhere in the world, where different privacy and data management laws might apply – simply put, customers cannot be 100% sure that local authorities will not have physical access to a server with their data stored on it. This is especially important for companies that do business across national boundaries. For example, the European Union places strict constraints on what personal data can be stored in the cloud, and for how long it can be stored there. Many banks and financial organisations also require customers' financial data

[4] These web sites aim to collect data on all reported incidents of cloud service outages and rank cloud providers accordingly. A representative example here is CloudHarmony – https://cloudharmony.com/status-1year-of-compute-group-by-regions-and-provider.

to remain within the borders of the home county. Obviously, in these circumstances many organisations would never be willing to release their sensitive data outside their corporate network, and, therefore, will not move the IT systems to the cloud.

- *Security* is the last, but admittedly the most crucial factor hindering the adoption of the cloud model. It is nowadays seen as the top factor preventing a company from migrating to the cloud, as indicated by various surveys [88, 90].IT security requirements need to be compliant with national and international IT processing standard agencies, taking into consideration various dimensions, including the following [58]:
  - Statutory compliance for regulations, laws and agency needs.
  - Characteristics of data for accessing those basic preservations needed by data set of an application.
  - Confidentiality and privacy for protecting against criminal and accidental access for information.
  - Integrity for ensuring that the data is complete, authorised, and accurate.
  - Data access and control policies for determining the location of data storage and users who have privilege for accessing data.
  - Governance for ensuring that the cloud service providers are necessarily transparent, supply requested information for agencies for independent and appropriate access, possess required monitoring, management and security controls.

The remote cloud-based model implies that customers inevitably have to send and store some of their data over the network to the cloud environment, which is not guaranteed to be protected against various kinds of threats and attacks. Customers need to be aware that there is always a possibility that data stored and processed remotely in the cloud can be compromised. Security is admitted by both industry and academia as the key challenge for the development of cloud computing, and a lot of efforts are currently being put into addressing these challenges.[5] Arguably, cloud computing will never become a truly ubiquitous model for delivering IT services until the issue of security is properly and successfully addressed. With the presented research work, we are also aiming at contributing to advancing the state of the art in this research area, as it will be further explained in the upcoming chapters of this document.

2.5 Summary

The goal of this chapter was to familiarise the reader with the main context of the presented research work – namely, Cloud Computing and an existing challenge of insufficient level of data privacy and security. The chapter presented an overview of cloud computing development and evolution. It also presented several existing definitions of cloud computing and outlined a list of main benefits associated with this emerging model of delivering IT services remotely over the network. Besides the main advantages, the chapter also listed potential shortcoming of migrating to the cloud environment, among which data privacy and security are seen as the main challenges. Accordingly, the material explained in this chapter will be further used throughout the rest of this document to explain the details of the proposed approach.

[5] Cloud Security Alliance, for example, is an organisation consisting of both industrial and academic partners, which collaborate to define and develop best practices to help ensure secure cloud computing environments.

Chapter 3

Access Control: a Review

Objectives:

- To provide an overview of general concepts and terminology of access control.
- To review access control in cloud computing and existing challenges.
- To conduct a literature survey on access control mechanisms in cloud computing.

3.1 Introduction

In the previous chapter, we familiarised the reader with the notion of cloud computing and main benefits making this novel model for delivering IT services attractive to the end user. Together with the potential benefits, we also summarised its potential shortcomings, among which security and data privacy are nowadays seen as the key factors preventing cloud computing from the ubiquitous adoption by enterprises and individual users.

As a possible way of addressing these challenges, we are developing a policy-based access control mechanism for cloud environments, which will be further explained and discussed in this thesis. To help the reader understand the contribution of our proposed approach and position it with respect to the overall body of existing work in the relevant research field, in this chapter we provide an overview of the state of the art in access control in cloud environments. We first brief the reader on the general concept of access control and then proceed with a literature survey aiming to provide a holistic and through review of the current state of the art and identify existing research gaps.

3.2 Access control: an overview

To date, the topic of security in computer systems has been attracting attention for several decades, and is still seen as one of the top challenges both for the academia and the industry [18]. Since the 1970s, when information systems started exponentially growing in size and complexity, an increasing number of applications served multiple users, which in turn led to an increased awareness of data security issues [81].

Computer systems security is typically considered as a highly complex and multi-faceted research discipline. Each of the sub-fields constituting the security research discipline (e.g., user authorisation, user authentication, auditing, administration, data privacy, etc.) is seen as independent and challenging research area in their own right. Another fundamental area underpinning the whole discipline of computer security is access control, which is the focus of the research work presented in this thesis. It is worth noting that these sub-disciplines (albeit they are demanding and pressing in their own right) are expected to be considered in a cross-cutting and interdependent manner. Security practitioners often see computer security is a chain. Accordingly, just as a chain is only as strong as its weakest link, computer security system is only as secure as its weakest component. In other words, a truly holistic solution addressing the security challenge is expected to consider all possible dimensions of this pressing problem. Accordingly, albeit access control and its associated challenges in relation to cloud computing will remain the main context of this presented research effort, we are inevitably seeing our proposed contribution as part of larger picture of computer security.

As outlined by Sandhu and Samarati [82], who were among the pioneers in the field of computer security, "access control constrains what a user can do directly, as well as what programs executing on behalf of the users are allowed to do. In this way access control seeks to prevent activity that could lead to breach of security". It is worth explaining the relation between access control and user authentication. The latter is responsible for correctly identifying the user, whereas access control is typically based on the assumption that the user has been already identified, and, therefore, its main responsibility is to decide whether this specific user is permitted to access particular resources. Thus, the effectiveness of the access control mechanism relies on and tightly coupled with the effectiveness of the authentication mechanisms [82]. In other words, if an identity of the user and the authentication mechanism are compromised, the access control component will in most cases consequently fail to prevent a potentially malicious intrusion.

Similarly, an access control mechanisms is also expected to be coupled with an appropriate *auditing* mechanism (also known as an accounting mechanism). In a broad sense, an auditing component is responsible for the 'post-mortem' analysis of all the user requests and activities taking place within the managed computer system. Auditing is typically achieved by logging all relevant events for the later analysis, and serves multiple purposes [82]:

- Potentially malicious users may be discouraged from attempting violations if they are aware of their requests being recorded and inspected.

- Auditing can analyse and detect violations (or patterns leading to violations) in users' behaviour.

- Auditing can analyse and detect potential flaws in the security system of the managed IT environment.

- Auditing is needed to ensure that authorised users act within their privileges and do not attempt to misuse their access rights.

These dependencies of the access control component on the authentication and auditing once again are intended to demonstrate the pressing requirement to design and implement security mechanisms in a systematic and holistic manner.

3.2.1 Access control policies and access control mechanisms

When implementing access control in IT systems, it is important to understand the two fundamental concepts – namely, policies and mechanisms. The former are usually seen as high-level guidelines, which serves to determine how user access is controlled and access granting decisions are taken [82].

On the other hand, access control mechanisms are typically implemented as low-level configurable software and hardware components, responsible for the enforcement of respective access control policies. Such mechanisms are expected to be policy neutral [70] – that is, to be independent of the various types of access control policies, for which they could be potentially used. Ideally, a truly policy neutral access control mechanism is able to be applied to a wide range of diverse and heterogeneous policies (i.e., defined using various formalisms and languages) in a transparent and seamless manner. Such a policy-agnostic behaviour can be achieved via a standardised and established approach to access control policies definition [44, 79].

In practice, however, it is more common to implement access control mechanisms which are re-usable across a limited set of security-related scenarios and respective access control policies. Since not all computer systems are characterised with the same level of the desired protection level, different policies may be suitable for different scenarios. For example, some systems dealing with sensitive data (e.g., applications processing banking, health, governmental data, etc.), are expected to follow most stringent access control policies, whereas in other systems such a high level is not required (or even not desired at all). These considerations suggest that the choice of appropriate access control policy depends on the particular characteristics of the IT environment to be protected, and in the next section we will examine the topic of access control in the context of cloud computing platforms – highly-distributed, remote and virtualised environments.

3.3 Access control in the cloud

The increasing popularity and adoption of cloud computing is still hindered with the ever-growing and pressing security challenges such as user management, access control, and data privacy [12, 15, 74, 91]. As opposed to traditional, in-premises IT systems, cloud environments rely on the virtualisation technology to support multi-tenancy and resource sharing. These aspects of cloud computing are seen as the key factors putting the security and data privacy in clouds at risk [89, 7].These features pose unique security and access control requirements due to sharing of underlying physical resources in a cloud environment among a large number of potentially untrusted tenants, which may in turn lead to an increased risk of side-channel attacks [76]. Moreover, cloud computations in the presence of multi-tenancy may result in interferences and, as a consequence, unauthorised information flows [7].

Furthermore, in the cloud, which is a networked and distributed environment, authentication and access control are taken to a different level of complexity due several reasons. First, of malicious users can observe network traffic they can potentially replay authentication protocols, pretending to be legitimate users [82]. Second, in the cloud environment it is often required that not only users, but computers on the network acting on their behalf need to mutually authenticate each other, which opens additional opportunities for various compromises and attacks.

It is also important to take into consideration the heterogeneity of services in cloud environments, which requires a differentiated approach – that is, supports varying degrees of granularity – when implementing an access control mechanism. Apart from preventing various risks associated with unauthorised use of cloud resources and services, such a mechanism would also serve as a basis for implementing standard security measures in the context of contemporary enterprise-level cloud ecosystems.

Heterogeneous cloud environments typically do not allow implementing a single authorisation mechanism, a single policy language or a single enforcement mechanism for customers, using several cloud providers or switching between private and public clouds [90]. Each cloud provider typically has its own access control mechanism, which is limited in its flexibility to support other mechanisms. In these circumstances, a desirable access control mechanism for the cloud is expected to work with all data (i.e., formats, languages, representations) regardless of where they are stored, and cloud users are expected to be able to manage their access policies to manage access to their respective data and resources, deployed in the cloud.

Taken together, the presented considerations suggest that cloud security in general, and user management and access control in particular, are non-trivial, complex, and challenging tasks, which require novel approaches to address them.

3.4 State of the art in access control policies

There are several models for implementing access control in computer systems, proposed in the literature. In this section, we will first brief the reader on these existing models providing some examples relevant to the context of the presented research. Then, we will specifically focus on the topic of access control in cloud environment and present several research works, which we find most relevant to our own work.

3.4.1 Existing access control models

Historically, the classification of access control models started with the first two, proposed by the Trusted Computer System Evaluation Criteria (TCSEC) published by the US Department of Defence [59] – namely discretionary access control and mandatory access control. However, since 1985, when these criteria were first published, the scientific community has developed a more fine-grained taxonomy of access control models, which we further discuss below. It is worth mentioning that the presented list s non-exhaustive and only includes models, which we find most prominent and relevant to our own work.

Discretionary access control

TCSEC [59] defines discretionary access control (DAC) in a high-level manner as “a means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject (unless restrained by mandatory access control)”. Simply put, DAC uses access control policies determined by the owner of the managed object, who is responsible to decide which users with which privileges are allowed to access that object [3].

There are two important concepts to be considered in the context of DAC:

- *Resource ownership* refers to the fact that every object (i.e., resource), such as files and data, has an owner. Typically, the initial owner of an object is actually a subject, which caused it to be created. Accordingly, it is the owner’s responsibility to define an access control policy for its object.

- *Access rights and permissions* are transferrable privileges for accessing specific resources, which can be assigned to third parties (i.e., other subjects), which can be assigned by the owner of those resources.

Mandatory access control

TCSEC [59] provides the following high-level definition of mandatory access control (MAC) – "A means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorisation (i.e., clearance) of subjects to access information of such sensitivity". In other words, MAC serves to allow access to a managed object only in the presence of pre-defined rules, which contain required information whether a particular user is allowed to access the resources [69].

As opposed to DAC, with a MAC-enabled security system, access control policies are centrally controlled by the security policy administrator, who are responsible for the implementation of organisation-wide security policies. In these circumstances, system ordinary users are not enabled to modify/over-ride the policy definition, whereas in DAC users may be in a position to introduce changes to the policy the ability to make policy and assign security attributes.

Rule-based access control

Rule-based (also known as label-based) access control [23, 99] can be seen as a sub-type of MAC, which serves to define specific conditions for accessing a specific resource in even more details. In its simplest form, a rule-based access control system can determine whether access to a particular object should be granted or not by matching sensitivity labels of the requesting subject and the requested object. If they match, the access is granted, otherwise it is denied.

Lattice-based access control

Lattice-based access control (LBAC) [80] is also seen as an instantiation of rule-based access control; it is a more complex model, which uses interactions between any combination of managed objects (e.g., files, computers, applications) and subjects (e.g., individual users, groups of users, organisations, etc.). In this case, the relationships between objects' levels of security and subjects' level of access can be mathematically expressed with partial order sets called as lattices. Simply put, a subject is only allowed to access an object if only if the security level of the subject is greater than or equal to that of the object.

Role-based access control

Role-based access control (RBAC) [81, 44, 79, 69] is nowadays seen as one of the most prominent and widely adopted models for implementing access control in a wide range of enterprise software systems. The basic principle in RBAC is that access control policies are defined by the managed system, not by the owner. In this sense, it is different from both DAC and MAC. On the one hand, as opposed to DAC, where users are typically allowed to control their resources, in RBAC this access is managed at the system level, which is beyond the user's control. On the other hand, albeit BAC is non-discretionary, it cannot be classified as MAC mainly due to the way permissions are handled and granted. More specifically, MAC controls individual access (i.e., read and write) permissions mainly based on the user's access rights and some additional labels, whereas in RBAC permission collections are involved, which may include more complex operations (e.g., a database transaction, which typically goes beyond the notion of a simple read/write access operation). Accordingly, such sets of permissions in RBAC can be seen as roles. There are three main principles underpinning the RBAC model:

- *Role assignment* refers to the fact that a subject can access or execute a transaction on an object only if it has been assigned a corresponding role.
- *Role authorisation* refers to the fact that a subject's active role must be appropriately authorised prior to accessing an object.

- *Transaction authorisation* refers to the fact that a subject can execute a transaction on an object, only if this type of transactions has been previously authorised for the subject's active role.

Besides these three principles, RBAC often also allows putting additional constraints and conditions. Moreover, roles in RBAC can be combined, thus forming role hierarchies, where higher-level roles subsume permissions owned by lower-level sub-roles.

Attribute-based access control

Attribute-based access control (ABAC) [50, 56] uses the concept of attributes of the user (rather than of the subject representing the user) to determine access rights. Accordingly, the user has to prove certain claims about his attributes so as to be granted access to a resource by the access control engine. The set of attribute claims to be proved is defined by attribute-based access control policies. A traditional example of an attribute-based access to some Web resources is to prove that the user is at least 18 years old.

Among other types of access control models it is worth mentioning some novel, non-traditional approaches such as intent-based access control [6], emotion-based access control [5], responsibility-based access control [42], and break-glass access control methods [20, 45].

3.5 Related works: a survey

As we have seen it in the previous section, there have been various access control models proposed in the literature over the years [81, 1, 14, 19, 78], which provided general theoretical foundations for the further development of the access control research. In the rest of this section we will brief the reader on research works in the field of access control policies, which we find relevant to our own work.

One of the early general formalisms for expressing authorisation and access control rules were proposed by Woo and Lam [98]. The proposed framework is based on the capabilities of default logic to support non-monotonicity during the formal reasoning process. Using default logic, it is possible to define the “default” behaviour of the access control system, without specifying additional exceptions. In situations when a new policy is added to an existing set of policies, it will be immediately picked up by the reasoner, and formerly-accessible subject may become inaccessible due to the newly-added rules. One of the potential shortcomings of the proposed approach (and default logic in general) is that reasoning over the set of policies might not eventually lead to a conclusion, thus resulting in situations when an authorisation request might not have an answer.

Jajodia et al. [52] in their work proposed a logical language for expressing authorisations called Authorisation Specification Language (ASL). The proposed ASL serves to specify access control policies and corresponding enforcement logic. It follows a fine-grained approach and allows specifying different policies on different objects, based on the specific requirements. ASL can be classified as a role-based access control language, as it enables users to specify different policies for different user roles and groups. The authors demonstrate with the expressivity of the proposed language (e.g., putting constraints on consistency and completeness) with a number of use cases. The authors claim that one of the main benefits of ASL is that it supports specification of multiple co-existing access control policies within the same managed system in the presence a dedicated component responsible for potential conflict resolution.

Jajodia et al. [51] moved on with their research and investigated potential ways of resolving inconsistencies and conflicts among authorisations. The proposed approach is based on rules, and the authors see flexibility as one of the main advantages of the approach.

This functionality is achieved by using the Flexible Authorisation Framework (FAF). One of the key components of the framework is a repository of policies, from where policies (and policy compositions) can be fetched and enforced in a concurrent manner. Among the potential shortcomings of the proposed approach is that the framework does not seem to support expressing such policy compositions natively and requires an additional language to perform this task. Also, the authors do not consider temporal constraints and dependencies as part of the functionality of the proposed framework.

This lack of the temporal dimension in specifying and enforcing access control policies was addressed by Bertino et al. [17, 16]. The authors developed a temporal model for access control, which is based on time intervals associated with authorisations to define their respective validity periods. Accordingly, rules in the proposed approach are defined by establishing temporal relationships between authorisations. On the other hand, however, the authors do not provide any evidence of support for policy compositions and handling multiple policies at a time in their approach.

The challenging topic of access control in distributed systems was addressed by Damianou et al. [28, 27, 29] and their proposed policy specification language Ponder. Ponder is a high-level, declarative, object-oriented language for managing access control policies in distributed systems. Its expressivity allows definition of a wide range of access control management actions, including authorisation, filtering, refraining, and delegation. Ponder supports policy compositions (including groups, roles, relationships and management structures). The object-oriented approach enables creation of user-defined types of policies and consequent instantiation of these policies with different parameters. Among the potential shortcomings of Ponder, the authors mention the insufficient support for concurrency constraints and the lack of formal semantics, which limits the possibilities of applying formal reasoning to policies defined with Ponder.

The Ponder approach was extended by Twidle et al. [92], who implemented a software framework for enforcing policies based on the specification of the original language Ponder. The current implementation of the framework mainly supports obligation and authorisation policies, as demonstrated by the authors. However, the proposed system still does not support formal semantics of the underlying language, and the reasoning processes are mainly handled by the general-purpose programming language in a hard-coded manner.

The eXtensible Access Control Markup Language (XACML) [9, 47, 60] was proposed by OASIS – a global consortium that works on the development, convergence, and adoption of standards for security and other challenging areas of IT. The proposed language is useful in the context of specifying access control policies of arbitrary complexity in various distributed systems. It is an open-source project based on XML, and therefore, can be easily extended to meet new emerging user requirements. As a uniform, standard and interoperable format, it also proves to be useful in bringing together multiple heterogeneous (legacy) systems. Being highly expressive, XACML, however, suffers from its complexity and verbosity. It seems to be difficult to implement software tools, which would fully support the expressivity and the semantic of the language. In other words, currently there seems to be no automated reasoner, which would be fully responsible for enforcing XACML-based policies.

Pugliese and Tezzi [73] were challenged by creating a simple, yet expressive alternative to XACML proposed by OASIS. In their work, the authors propose Simple Access Control Policy Language (SACPL) – an expressive language for defining access control policies in various contexts. It follows the attribute-based access control model, and its main advantage, as indicated by the authors, is its relative simplicity and the seamless integration with Service Component Ensemble Language (SCEL) [36, 37], which is a language, designed to program autonomic components and their respective interactions, while supporting formal reasoning activities over their behaviour.

Siewe et al. [85] developed a compositional framework for the specification of access control policies, based on Interval Temporal Logic (ITL). Using ITL formulas it is possible to define authorisation and delegation rules, both positive and negative. The authors distinguish between signed rules, which are used for actual definition of access rights, and enforcement rules, which define underlying policy enforcement logic. The proposed approach is also equipped with conflict resolution capabilities – and contradictions and conflicts can be handled appropriately by the corresponding enforcement rules. One of the main benefits of the proposed approach, is the use of ITL as the underlying formalism and its support for specifying both functional and temporal properties of the managed resources. In conjunction with the SANTA framework, the authors also demonstrate the viability of the proposed approach.

3.5.1 Access control policies in cloud environments

Takabi et al. [90, 89] in their work address the challenge of access control in cloud environments and propose a semantic-based approach to policy management. The authors propose using Semantic Web languages (i.e., Web Ontology Language – OWL, Semantic Web Rule Language – SWRL) to define and reason over security policies. Among the main benefits of this approach authors list the reasoning power of OWL and SWRL, which are based on Description Logics. As a positive side effect of using Description Logics, the authors claim that OWL-defined access control policies can be seamlessly translated to other policy languages and formalisms. Another important benefit of applying the Semantic Web languages in the context of cloud-based access control is interoperability and possibilities to apply same policies across a multiple cloud platforms. Human-readability, re-usability, extensibility is also among the potential advantages of the proposed approach.

A similar semantic approach is presented by Moreau et al. [65], who introduced an extension to the KAoS ontology [94, 93] to capture the domain of access control in cloud and grid environments. The KAoS approach is based on an OWL ontology (and, therefore, on Description Logics), and thus can benefit from high expressivity and formal reasoning. Using an ontology as an existing established vocabulary of terms is another benefit of the KAoS-based approach. The main contribution of the authors is their extension of the core ontology and policies to manage the behavioural specification of grid registries.

Hu et al. [49] in their work also follow the trend of using Semantic Web languages to create an access control approach for cloud environments. As highlighted by the authors, main benefits of doing so are clear semantics and established vocabulary, support for formal reasoning and semantic interoperability. The proposed distributed role-based access control approach uses an ontology-based Semantic Access Control Policy Language (SACPL). The approach is inspired by XACML – it uses its basic concepts (i.e., subject, object, action and attribute variables) and also introduces some new concepts (i.e., priority and confidentiality). The authors claim that semantic interoperability of the proposed approach is the key advantage of the approach, which is a key requirement in highly heterogeneous cloud environments. On the other hand, as a potential shortcoming of the approach we can identify the lack of a policy conflict resolution mechanism.

3.6 Analysis and discussion

In the previous section, we surveyed the relevant literature with a goal to identify existing limitations and gaps, which would be address by our own proposed approach. To this end, we can draw several conclusions:

I. Cloud computing, as a relatively new research discipline, poses new emerging requirements for the security researchers to be addressed. Among the most pressing concerns are access control and user management. As indicated by our survey, most of the surveyed authors do not explicitly consider cloud platforms as a potential application scope for their respective approaches, simply because cloud were not there yet, when the papers were published. Despite this lack of explicit 'cloud-orientation', most of the surveyed approaches still seem to be unsuitable due to the fact that they were not designed and implemented to be applied in a distributed environment. They seem to be succeeding in managing access rights locally in relatively small-scale, localised environments, but there is little evidence that they also handle access control in the presence of multi-tenancy and resource sharing, increased concurrency, virtualisation, etc. – the key features of cloud computing.

II. The dynamic nature of the cloud, its increased multi-tenancy and ever-growing number of users, simultaneously accessing cloud resources, require that the access control component is capable of handling the historical and temporal aspects of the user access. As the majority of surveyed approaches tend to neglect the temporal dimension, for a cloud-based access control system it is essential to be able to put time constraints when defining and enforcing access control policies. Moreover, it is also important to keep track of the whole history of access in the cloud, including such information as access attempts, modification of user access rights, transactions on resources, etc. To a certain extent, this can be seen as a 'stateful' approach, which also takes into consideration not just the current state (i.e., a 'stateless' approach), but also all the information concerning what has actually led to this state over some period of time.

III. The third observation is the trend of utilising Semantic Web technologies in the context of access control in the cloud. This can be explained by several reasons [33, 32]. First, due to the high heterogeneity of (multi-) cloud environments, it is important to use a standardised vocabulary of terms (i.e., OWL ontologies). Second, due to the formal semantics of OWL and SWRL, access control administrators can benefit from software reasoners – already-existing, optimised and reliable enforcement mechanisms. Third, OWL ontologies, once implemented and published, facilitate re-usability and interoperability, as the same access control policies defined with OWL and SWRL can be applied across several cloud platforms. Fourth, the Semantic Web languages support object-orientation, thus it is possible to create hierarchies of roles, rules, attributes, resources, etc. The last but not the least, the Semantic Web languages are human-readable, which makes them easier to work with even for non-professional programmers or computer scientists.

Taken together, these observations outline the main functional requirements for our own proposed approach, which can be summarised as follows:

I. *Support for the spatial dimension and cloud-orientation*: the envisaged access control approach has to be designed and implemented with support for several cloud-intrinsic features in mind, such as distributed architecture, multitude of network locations, virtualisation, multi-tenancy, shared resources, increased number of users accessing resources, etc.

II. *Support for the temporal dimension*: the envisaged approach has to be capable of capturing the history of access, as well as putting and enforcing temporal constraints in access control policies.

III. *Formal semantics and interoperability*: these are the two keys aspects of the Semantic Web-based approaches for access control. Accordingly, our envisaged approach has to inspire from the Semantic Web techniques to demonstrate these key features, as well as some other of its minor positive aspects.

3.7 Summary

In this chapter, we familiarised the reader with the existing research efforts in the field of access control – one of the most challenging tasks in engineering secure computer systems.

First, we introduced the general concepts of access control, such as polices and enforcement mechanisms. Then the chapter proceeded with outlining main challenges for access control in cloud environments – namely, these are resource sharing, multi-tenancy, increased number of users, distributed architecture, etc. Then we presented a brief literature survey of the existing relevant research work on access control (including access control in the cloud). Finally, we provided an analysis of the surveyed approaches, which helped us to identify existing gaps and outline requirements for our own research work.

Chapter 4

Formal Underpinnings of the SANTA Policy Language

Objectives:

- To introduce the SANTA policy languages, which serves as a baseline for our research and contributions.
- To explain the application scope of the SANTA policy language: what this language is and what it is used for.
- To introduce and explain Interval Temporal Logic, which acts as a formal underpinning of the SANTA language.
- To outline an existing gap of insufficient support for location-aware data privacy in the SANTA language.
- To introduce Topological Logic and explain its application to SANTA to enable location-aware access control and data privacy.

4.1 Introduction

The main goal of this chapter is to familiarise the reader with the SANTA policy language – i.e., its theoretical underpinnings, formal semantics, application scenarios, and limitations. Explaining these notions is important, as the SANTA language is taken as the main baseline in the context of the presented PhD research – that is, by introducing the current state of the SANTA language, we aim to outline an existing gap to be addressed by the present work.

Accordingly, the material explained in this chapter is three-fold. First, it introduces the SANTA language at its current state and briefs the reader on the expressivity of the language – that is, in what kind of use case scenarios using the SANTA language is convenient and efficient. Second, the chapter outlines an existing limitation of the language – namely, insufficient support for handling location-aware data privacy. Third, it introduces Topological Logic as a way of adding location-aware features to the core design of the SANTA language (which are yet to be explained in the next chapters of this thesis).

4.2 SANTA policy language: an overview

SANTA is a formal language for expressing access policies in complex computer systems [85, 55, 53, 54]. It follows a rule-based approach (as explained in the classification in the previous chapter) to specify (sets of) access control rights in terms of policy rules and their compositions. A simple access control policy, defined by SANTA, typically consists of three types of rules:

1. *Positive authorisation rules* – these are rules, which define if a subject is allowed to access an object;

2. *Negative authorisation rules* – these are rules, which define if a subject is restricted from accessing an object;

Table 4.1 Basic syntax of the SANTA policy language.

Subject: $su ::= \mathbf{S}_i \mid cs$
Object: $ob ::= \mathbf{O}_i \mid co$
Action: $ac ::= \mathbf{A}_i \mid ca(e_1,...,e_n)$
Premise of a rule: $pr ::= pr_1 \; ; pr_2 \mid pr_1$ **and** $pr_2 \mid pr_1$ **or** $pr_2 \mid$ **always** $pr \mid$ **sometime** $pr \mid$ **not** $pr \mid$ **if** be **then** pr_1 **else** $pr_2 \mid$ **exists** x *in* $se : pr \mid$ **forall** x **in** $se : pr \mid$ **last**$(e) : pr \mid e : pr \mid be$
Rule: $ru ::= [rn ::]$ **allow** (su, ob, ac) **when** $pr \mid$ $[rn ::]$ **deny** (su, ob, ac) **when** $pr \mid$ $[rn ::]$ **decide** (su, ob, ac) **when** pr
Rule: $po ::= \; ru_1 \; ... \; ru_2 \mid$ **policy** $pn :: po$ **end** $\mid$ $po_1 \; ; po_2 \mid$ **aslongas** $be : po \mid$ **unless** $be : po \mid$ $e : po \mid$ **if** be **then** po_1 **else** $po_2 \mid$ **repeat** po

3. *Decision rules* – these rules serve to resolve possible conflicts, resulting from specification of the previous two types of rules.

These simple rules serve as atomic ‘building blocks’ for defining access control policies. They can then be combined into composite policies. In the following, we provide the syntax of access control policies and examples of their usage.

Table 4.1 contains the basic syntax of the SANTA policy language, where:

e is an expression;

be is a Boolean expression;

se is a set expression with usual support for operators and semantics;

$\mathbf{S}_i$ is a subject variable, where i is an arbitrary name;

$\mathbf{O}_i$ is an object variable, where i is an arbitrary name;

$\mathbf{A}_i$ is an action variable, where i is an arbitrary name;

pn is a name for a policy;

rn is an (optional) name for a rule.

This brief introduction to the SANTA policy language is further widened in the next section, which introduces and explains SANTA's main underpinning logical formalism known as Interval Temporal Logic. Even more further details on the specifics of ITL can be found in [54, 24].

4.3 Underpinning formalism: Interval Temporal Logic

Interval temporal logic (ITL) (sometimes also referred to as interval logic) is a flexible notation for representing both propositional and first-order logical reasoning about periods of time in the context of hardware and software systems [4, 67]. First-order interval temporal logic was initially formulated in the 1980s for the specification and verification of hardware protocols. It is a subset of a more general subset of temporal logics, which was originally developed by Moszkowski [66]. It is useful in the formal description of hardware and software for computer-based systems whenever there is a requirement to capture the temporal dimension – for example, keep historical track of events, which took place within the system.

One of the main features of ITL is its compositionality, which is seen as a considerable benefit and an issue at the same time. This feature allows ITL to handle both sequential and parallel composition of atomic rules (unlike the other types of temporal logic). Moreover, instead of dealing with infinite sequences of system states, ITL deals with finite sequences. ITL offers powerful and extensible specification and proof techniques for reasoning about system properties involving safety, liveness and projected duration [68]. These rich expressivity and

Table 4.2 Syntax of Interval Temporal Logic.

Expressions: $e ::= z \mid a \mid A \mid g(e_1,...,e_n) \mid \circ v \mid \mathbf{fin} v$
Formulae: $f ::= true \mid q \mid Q \mid h(e_1,...,e_n) \mid \neg f \mid f_1 \wedge f_2 \mid$ $\forall\, v\ \bullet f \mid \mathbf{skip} \mid f_1\ ;\ f_2 \mid f*$

flexibility, however, come at a cost – there is a pressing requirement of handling and resolving potential conflicts when two or more simple rules are combined into a compositional policy.

The fundamental concept of ITL is an interval. An interval σ is considered to be an (in)finite sequence of states σ_1, σ_2, σ_3, ..., where a state σ_i is a mapping from the set of variables Var to the set of values Val. The length $|\sigma|$ of an interval σ_1, σ_2, σ_3, ... is equal to n – that is, one less than the number of states in the interval. This latter feature means that the length of a single-state interval is equal to 0.

The basic syntax of ITL is represented in Table 4.2, where:

z is a constant integer value,

a is a static integer variable (non-changeable within an interval),

A is a state integer variable (changeable within an interval),

v is a static or state integer variable,

g is an integer function symbol,

q is a static Boolean variable (non-changeable within an interval),

Q is a state Boolean variable (changeable within an interval),

h is a predicate symbol.

4.3.1 Informal semantics of the main ITL constructs

We now consider in more details the informal semantics of the key ITL constructs, presented in the previous section.

- **skip**: The skip operator always succeeds and consumes (defines) exactly one cycle – that is, it is a unit interval with the length equal to 1 (i.e., an interval between two states).

- f_1 ; f_2: this operator holds if an interval can be decomposed into a prefix and a suffix intervals, such that f1 holds over the prefix and f_2 holds over the suffix, or, alternatively, if the interval is infinite and f_1 holds for the whole interval. Note the last state of the interval σ_k, over which f_1 holds is shared with the interval over which f_2 holds, as depicted in Figure 4.1.

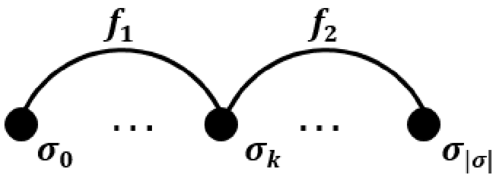


Fig. 4.1 Informal semantics of the operator f_1 ; f_2.

- $f*$: this operator holds if the interval is decomposable into a finite number of intervals, so that for each of them f holds, or the interval is infinite and can be decomposed into an infinite number of finite intervals, so that for each of them f holds as well. This operator is illustrated in Figure 4.2.

- v: this represents the value of v in the next state, when it is evaluated on an interval with a length equal to or greater than 1 (i.e., at least one). Otherwise, it takes an arbitrary value.

- **fin**v: this represents the values of v in the final state, when it is evaluated on a finite interval. Otherwise, it takes an arbitrary value.

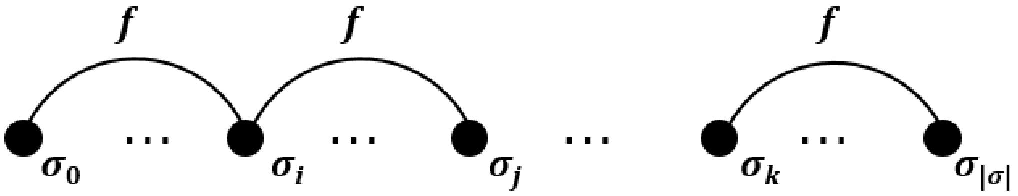


Fig. 4.2 Informal semantics of the operator $f*$.

4.3.2 Derived constructs

Apart from the core elements of the ITL syntax we can also derive several additional constructs, which serve to make the writing of ITL expressions more natural and simpler. In particular, a (non-exhaustive) list of the most important derived constructs is presented and explained below.[1] These will be further used in the upcoming chapters of this thesis. Table 4.3 lists some of the derived constructs used in the remainder of this research.

4.3.3 Policy-level information flow analysis

As explained above, in the context of access control a subject requests to perform an action upon an object. Subjects, objects, and actions are the key concepts of access control in general, and the SANTA language in particular. Accordingly, an access control policy governs the process of accessing objects by subjects.

In this respect, we can distinguish between two main categories of actions, which can be allowed or denied by the policy enforcement engine:

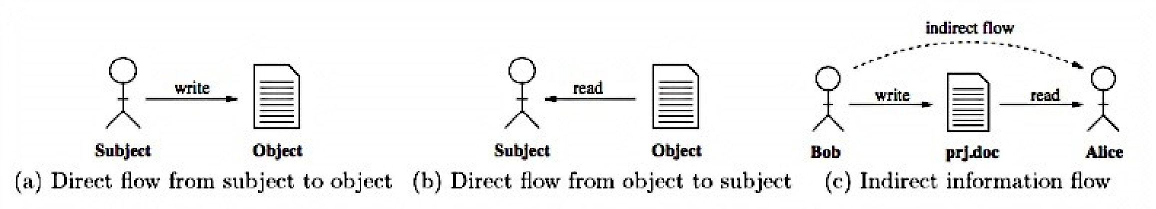
- Read actions (see Figure 4.3b), which transfer information from the requested objects to the requesting subject. For example, checking the current available balance on a bank account or reading file contents on disk transfers information from the requested bank account or file to the subject that executes the read action.

[1] Please note that the basic Boolean operators $\wedge$ (logical adjunction), $\vee$ (logical disjunction), $\neg$ (logical negation) and $\supset$ (logical implication) are derived as usual.

Table 4.3 Derived constructs.

Derived construct	Explanation
$\circ f \hat{=} \textbf{skip} \; ; f$	**Next** f: f holds from the next state. **Example**: $\circ(X = 1)$ – any interval such that the value of X in the second state is 1 and the length of that interval is at least 1.
$\textbf{more} \hat{=} \circ \textbf{true}$	**Non-empty interval**: any interval with the length equal to at least 1.
$\textbf{empty} \hat{=} \neg \textbf{more}$	**Interval**: any interval with the length equal to 0 (i.e., only one state).
$\textbf{inf} \hat{=} \textbf{true} \; ; \textbf{false}$	**Infinite interval**: any interval with an infinite length.
$\textbf{finite} \hat{=} \neg \textbf{inf}$	**Finite interval**: any interval with a finite length.
$\diamond f \hat{=} \textbf{finite} \; ; f$	**Sometimes** f: any interval such that f holds over a suffix of that interval. **Example**: $\diamond X \neq 1$ – any interval such that there exists a state, in which X is not equal to 1.
$\Box f \hat{=} \neg \diamond \neg f$	**Always** f: any interval such that f holds for all suffixes of that interval. **Example**: $\Box(X = 1)$ – any interval such that the value of X is equal to 1 in all states of that interval.
$\boxed{\text{i}} f \hat{=} \neg(\neg f; \textbf{true})$	**Box-i**: any interval such that f holds over all prefix sub-intervals.
$\boxed{\text{a}} f \hat{=} \neg(\textbf{finite} \; ; \neg f \; ; \textbf{true})$	**Box-a**: any interval such that f holds over all sub-intervals.
$\textbf{fin} f \hat{=} \Box(\textbf{empty} \supset f)$	**Final state**: any interval such that f holds in the final state of that interval.
$\exists v \cdot f \hat{=} \neg \forall v \cdot \neg f$	**Existential quantification.**
$f^n \hat{=} \begin{cases} false & \text{if } n < 0 \\ empty & \text{if } n = 0 \\ f \; ; f^{n-1} & \text{if } n > 0 \end{cases}$	f **repeats** n **times.**
$\textbf{len}(e) \hat{=} \textbf{skip}^e$	The statement holds if the length of the interval is e.

- Write actions (see Figure 4.3a), which transfer information in the opposite direction – that is, from the requesting subject to the requested object. For example, depositing a bank account or appending records to a file.
- There are also actions, which do not belong to these two types. This typically means that there is no direct interaction between the subject and the object, and information flows indirectly via, for example, a shared file (see 4.3c). Moreover, there are also actions, which can be classified as both read and write actions – that is, information flow is bi-directional.



(a) Direct flow from subject to object (b) Direct flow from object to subject (c) Indirect information flow

Fig. 4.3 Direct and indirect information flows: (a) Direct flow from subject to object; (b) Direct flow from object to subject; (c) Indirect information flow [54].

Accordingly, we employ the following definitions for direct information flows, allowed by the SANTA enforcing engine ($Actions_r$ and $Actions_w$ are subsets of the overall set of $Actions$, which represent all read actions and all write actions respectively.):

- An allowed direct information flow from a subject s to an object o takes place if the subject s is allowed to perform a write action on the object o [54]. This is illustrated below:

$$s \rightsquigarrow o \triangleq \bigvee_{\forall \alpha \in Actions_w} \mathbf{Aut}(s, o, a)$$

- An allowed direct information flow from an object o to a subject s, if the subject s is allowed to perform a read action on the object o [54]. This situation is illustrated as follows:

$$o \rightsquigarrow s \mathrel{\hat{=}} \bigvee_{\forall \alpha \in Actions_r} \mathbf{Aut}(s, o, a)$$

4.4 Policy rules

A SANTA rule typically expresses a single security requirement and acts as a basic 'building block' for constructing a more complex security policy. Every rule includes two parts – namely, the body (i.e., the *premise*) and the head (i.e., the consequence). The premise of a rule describes a set of system behaviours, which lead to the consequence that represents an assertion on the current system state, such as allowing or denying a particular access [54]. The consequence of a rule defines the decision taken by the reference monitor – a software component responsible for evaluating and enforcing access control policies within the managed computer system. The set of system behaviours in the premise is matched against the history of the system execution. It follows that it is possible to refer to sequences of previously observed system execution states, which underpins the possibility of the SANTA language to express history-based policies [2], and dynamic separation of duty constraints [78]. As far as events are concerned, we can distinguish between two types of events that can be referred to in the premise of rules – they are either those defined in the computational model, or external events that are observable by the reference monitor (as depicted in Figure 4.4).

Another important notion in the context of access control is authorisation. Authorisation is responsible for granting access to resources in the system. It defines whether execution

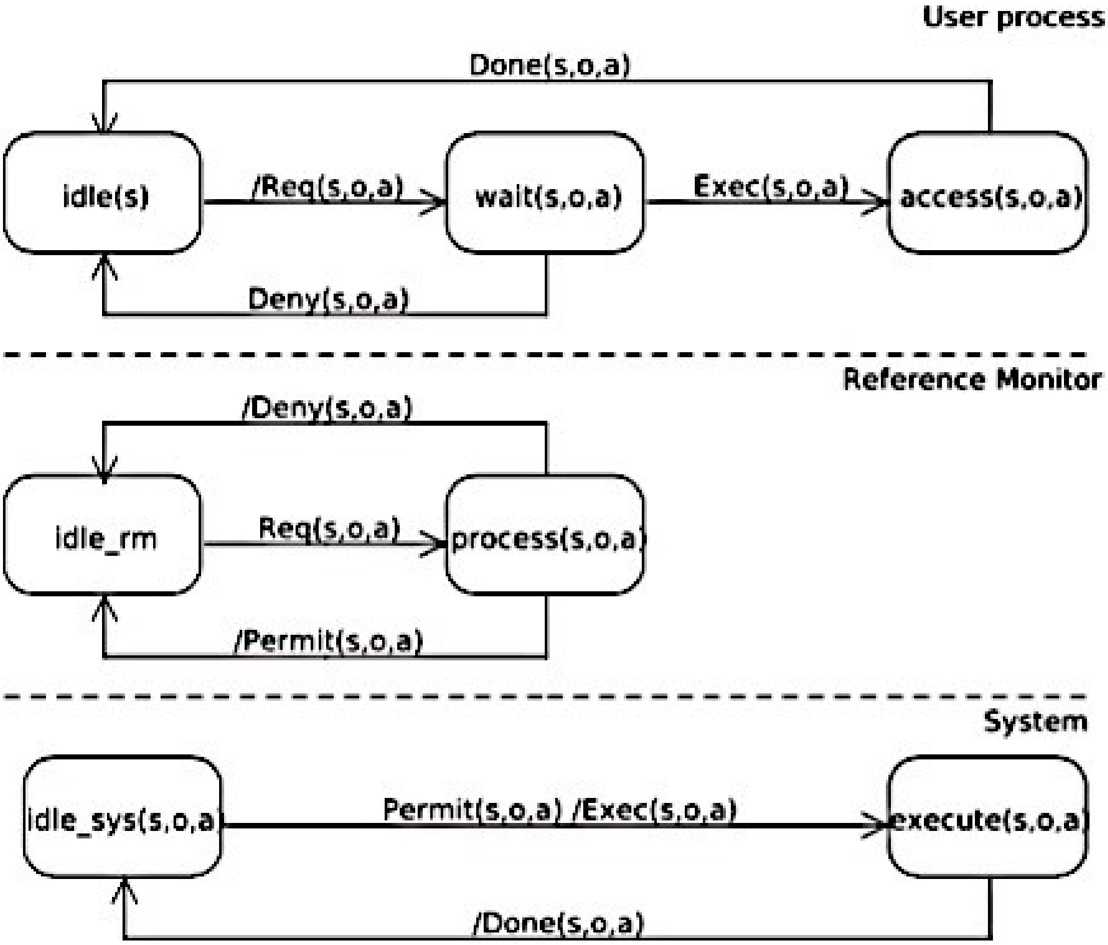


Fig. 4.4 Computational model of the SANTA rules [54].

of an action can be permitted with respect to the computational model. An authorisation rule defines possible conditions, under which a subject is allowed to perform an action on an object.

4.5 Expressivity and application scope of SANTA

In this section, we explain the most common scenarios, in which using SANTA for enabling access control is expected to prove to be useful. We demonstrate and explain simple examples [54], which are intended to *i)* showcase the capabilities and the application scope of the SANTA language, and *ii)* identify an existing gap – that is, an evidence, demonstrating that SANTA at its current state is not particularly suitable to address a particular situation. This latter shortcoming of SANTA in its turn will be addressed by our own research, as explained in the next chapters of the thesis.

The presented examples are based on several use cases, in which the issue of access control has been regarded as a pressing concern. These use cases, for example, include an electronic paper submission (EPS) system – a commonly used system, employed by universities and other academic institutions, to handle paper assignments submitted by students. To avoid any kinds of cheating and unauthorised access to the submitted materials it is important that the EPS system is equipped with a sufficient access control mechanism. Another use case is an e-banking system, where users are not allowed to perform unauthorised operations. A use case, which is based on a procurement system, involves a requirement of not releasing too much information to competing contractors so as not to create an advantage for any of them and facilitate fairness. The examples in Table 4.4 demonstrate how the SANTA policy engine can address these challenges.

Example 1: Unconditional authorisation (positive and negative). The example represents a situation when everyone can register a submission with the EPS system. In the example, **S** is a subject variable, which represents any subject (i.e., everyone submitting a paper). The object of the access control rule in this case is the EPS system itself (*eps*), which is referred to by its name ($eps \in Objects$). The action register has a parameter **O**, which is a known object in the system, expressed by an object variable. The first part of the example represents a positive authorisation rule. Negative authorisation rules (i.e., denial of access) are expressed in a similar way with the only difference that the rule consequence contains the keyword **deny**, as illustrated by the second part of the example.

$r_1 :: \textbf{allow}(\mathbf{S}, eps, register(\mathbf{O}))\ \textbf{when}\ true$

$r2 :: \textbf{deny}(\mathbf{S}, eps, register(\mathbf{O}))\ \textbf{when}\ true$

Example 2: Putting a condition on the current state of the system. This represents a situation when only the owner of a specific bank account can withdraw money from it. The condition is checked in the current state of the system. This is forced by the $e:$ construct – it forces the rule premise to be evaluated over the e long suffix of the execution history (i.e., over the history of states preceding the current state of the system). In case e equals to 0, the evaluation is performed only against the current state of the system.

$r_1 :: \textbf{allow}(\mathbf{S}, \mathbf{O}, withdraw())\textbf{when}\ 0 : owner(\mathbf{S}, \mathbf{O})\ \textbf{and}\ account(\mathbf{O})$

Example 3: Putting a condition on the history of system states. This example states that the subject must not perform an action on one and the same object more than once. Once an action has been performed on the subject, all further attempts will be denied. The condition is evaluated over the whole execution history. The construct sometime checks whether in any suffix of the history of the system states, the requested action has already be executed. Accordingly, **done(S,O,A)** represents an event generated by the system, when an action has been successfully authorised and performed. Please note that the same variable names **S, O,** and **A** in the rule consequence and the premise are explicitly used to denote that the very same subject, object, and action are concerned.

$r_1 :: \mathbf{deny}(\mathbf{S},\mathbf{O},\mathbf{A})\mathbf{when\ sometime\ done}(\mathbf{S},\mathbf{O},\mathbf{A})$

Example 4: Putting a condition on the history of system states. This example elaborates on the previous one. It represents a situation, when a subject S_1 is denied to perform an action if this action has already been performed by another subject S_2.

$r_1 :: \mathbf{deny}(\mathbf{S_1},\mathbf{O},\mathbf{A})$ **when sometime done**$(\mathbf{S_2},\mathbf{O},\mathbf{A})$ **and** $\mathbf{S_1} <> \mathbf{S_2}$

Example 5: An invariant. This example further explain the previous one, and represents a situation when a person is only allowed to take a loan at a bank, if he/she has never been bankrupt.

$r_1 :: \mathbf{allow}(\mathbf{S},\mathbf{O}_{loan},take)$ **when always not** $bankrupt(\mathbf{S})$

Example 6: Making choice. This example illustrates a situation when a child younger than 10 years old has to be given consent by both parents. If the child is older, consent given by only one parent may also suffice. Two distinct subjects $\mathbf{S_1}$ and $\mathbf{S_2}$ represent parents of the child S. The rule itself consists of two branches – if the child is younger than 10 years old, then the first branch is evaluated, which checks whether both parents have given consent before. The else branch uses the or statement to express that only prior consent of one parent is enough.

r_1 :: **allow**$(\mathbf{S},\mathbf{O},\mathbf{A})$ **when** $\mathbf{S_1} <> \mathbf{S_2}$ **and** $parent(\mathbf{S_1},\mathbf{S})$ **and** $parent(\mathbf{S_2},\mathbf{S})$ **and if** $age(\mathbf{S}) < \mathbf{10}$
then sometime done$(\mathbf{S_1},\mathbf{O},consent(\mathbf{A}))$ **and sometime done**$(\mathbf{S_2},\mathbf{O},consent(\mathbf{A}))$
else sometime done$(\mathbf{S_1},\mathbf{O},consent(\mathbf{A}))$ **or sometime done**$(\mathbf{S_2},\mathbf{O},consent(\mathbf{A}))$

Example 7: Collaboration. This example represents a situation when a door can be opened only if there have been at least two distinct subjects in the last five time units (i.e., system states), which requested the door to be opened. In these circumstances, two subjects need to collaborate in order to open the door within a limited time frame (5 last system states). It is worth noting that the outcome of these requests is not taken into consideration when deciding whether the door has to be opened or not – that is, the request event itself is important, and even if the two requests have been denied, the condition is still satisfied. The order of the requests also does not matter – moreover, the requests might have been made simultaneously.

r_1 :: **allow**$(\mathbf{S},door,open)$ **when** 5: **sometime req**$(\mathbf{S_1},door,open)$ **and**
sometime req$(\mathbf{S_2},door,open)$ **and** $\mathbf{S_1} <> \mathbf{S_2}$

Example 8: Time. This example represents a situation when a subject is not allowed to access the same resource within the last 10 time units. The example assumes that the current system time is **T**, time is treated as a set **TIME**, and existential quantification is used to bind the moment of the last access $\mathbf{last(1)} : \mathbf{done}(\mathbf{S},\mathbf{O},\mathbf{A_1})$ has taken place to $\mathbf{t_{last}}$. The comparison between this last access time and the current system time is performed with this statement $\mathbf{0} : \mathbf{t_{last}} + \mathbf{10} < \mathbf{T}$. Please note that the last access action may differ from the current request – i.e., $\mathbf{A_1}$ is not necessarily the same with **A**.

r_1 :: $\mathbf{deny}(\mathbf{S},\mathbf{O},\mathbf{A})$ **when exists** $\mathbf{t_{last}}$ **in** $TIME$:
$(\mathbf{sometime\ last}(1)\ :\ \mathbf{done}(\mathbf{S},\mathbf{O},\mathbf{A_1})$ **and** $\mathbf{t}_{last} = \mathbf{T})$ **and** $\mathbf{0} : \mathbf{t}_{last} + 10 < \mathbf{T}$

Example 9: Cardinality. The example represents a situation when a resource should not be accessed by the same subject for more than seven times.

r_1 :: $\mathbf{deny}(\mathbf{S},\mathbf{O},\mathbf{A})\mathbf{whensometimelast}(7) : \mathbf{done}(\mathbf{S},\mathbf{O},\mathbf{A_1})$

Example 10: Cardinality and time. These examples combine the two previous examples. It represents a situation when a subject should not be allowed to make more than consecutive 100 requests to access an object within the last 24 time units.

r_1 :: $\mathbf{deny}(\mathbf{S},\mathbf{O},\mathbf{A})$ **when exists** $\mathbf{t}_0$ **in** $TIME$:
$(\mathbf{sometime\ last}(100) : \mathbf{done}(\mathbf{S},\mathbf{O},\mathbf{A_1})$ **and** $\mathbf{t_0} = \mathbf{T})$ **and** $\mathbf{0} : \mathbf{t}_0 + 24 < \mathbf{T}$

Example 11: Sequential access. This example represents a situation when an invoice $\mathbf{O}_{inv}$ has to be received and authorised first, before it can be paid. This example entails a sequence of two distinct actions.

$r_1 :: \mathbf{deny}(\mathbf{S}, \mathit{bank}, \mathit{pay}(\mathbf{O}_{\mathit{inv}}))\ \mathbf{when\ not}\ (\mathbf{sometime\ done}(\mathbf{S}, \mathbf{O}_{\mathit{inv}}, \mathit{receive})\ ;\ \mathbf{sometime}$ $\mathbf{done}(\mathbf{S_A}, \mathbf{O}_{\mathit{inv}}, \mathit{authorise}))\ \mathbf{and}\ 0 : \mathit{role}(\mathbf{S}_A, \mathit{accountant})\ \mathbf{and}\ \mathbf{S}_A <> \mathbf{S}$

Example 12: Decision rule. This example represents a situation when the decision whether to grant access to a resource has to be taken after the decision whether it has to be denied. The example illustrates how potential conflicts, resulting from applying positive and negative authorisation rules in parallel, can be resolved.

$r_1 :: \mathbf{decide}(\mathbf{S}, \mathbf{O}, \mathbf{A})\ \mathbf{when}\ 0\text{: }\mathbf{allow}(\mathbf{S}, \mathbf{O}, \mathbf{A})\ \mathbf{and\ not\ deny}(\mathbf{S}, \mathbf{O}, \mathbf{A})$

Example 13: Closed decision rule. This example represents the same situation with a closed decision approach applied. The rules states that an action is denied unless explicitly allowed.

$r_1 :: \mathbf{decide}(\mathbf{S}, \mathbf{O}, \mathbf{A})\ \mathbf{when}\ 0\text{: }\mathbf{allow}(\mathbf{S}, \mathbf{O}, \mathbf{A})$

Example 14: Open decision rule. Similarly, this example represents the same situation with an open decision approach applied. The rules states that an action is allowed unless explicitly denied.

$r_1 :: \mathbf{decide}(\mathbf{S}, \mathbf{O}, \mathbf{A})\ \mathbf{when}\ 0\text{: }\ \mathbf{not\ deny}(\mathbf{S}, \mathbf{O}, \mathbf{A})$

Example 15: A history-based decision rule. This examples represents the same situation with a history-based decision taking applied. The example demonstrates that a decision can only be taken, provided there have been no access denials within the last 10 time units. In other words, an action is allowed if there have been no negative authorisations within the last 10 time units.

$r_1 :: \mathbf{decide}(\mathbf{S}, \mathbf{O}, \mathbf{A})\ \mathbf{when}\ 10\text{: }\ \mathbf{always\ not\ deny}(\mathbf{S}, \mathbf{O}, \mathbf{A})$

Table 4.4 SANTA expressivity examples.

The presented examples demonstrated definition of history-based policy rules using the SANTA language. In the next section, we will further explain SANTA's capabilities by illustrating how these 'building blocks' can be composed into policies in two ways – i.e., sequentially and in parallel.

4.6 Policies and compositions

Using SANTA, atomic rules can be combined to produce simple policies as follows:

policy p ::

$\mathbf{allow}(\mathbf{S}, \mathbf{O}_{loan}, take)$ **when always not** $bankrupt(\mathbf{S})$

$\mathbf{deny}(\mathbf{S}, \mathbf{O}, \mathbf{A})$ **when sometime last**$(7)\ :\ \mathbf{done}(\mathbf{S}, \mathbf{O}, \mathbf{A_1})$

$\mathbf{decide}(\mathbf{S}, \mathbf{O}, \mathbf{A})$ **when**

$0 : \mathbf{allow}(\mathbf{S}, \mathbf{O}, \mathbf{A})$ **and not** $\mathbf{deny}(\mathbf{S}, \mathbf{O}, \mathbf{A})$

end

As it follows from this example, a simple policy p is composed of atomic rules. In SANTA, all rules constituting a simple policy apply simultaneously. The policy p, in its turn, can be further used to combine more complex policy compositions. A simple policy typically serves to capture certain security aspects of a specific situation – that is, it is not generic and flexible enough to apply across a wide range of scenarios, which might emerge in the context of access control in dynamic and complex computer systems. Accordingly, to address more complex scenarios, simply policies are expected to be combined into policy compositions. As a result, the hierarchy of the SANTA policy compositions is the following – atomic rules are used to define simple policies, and simple policies are used to construct complex policy compositions.

4.6.1 Sequential composition

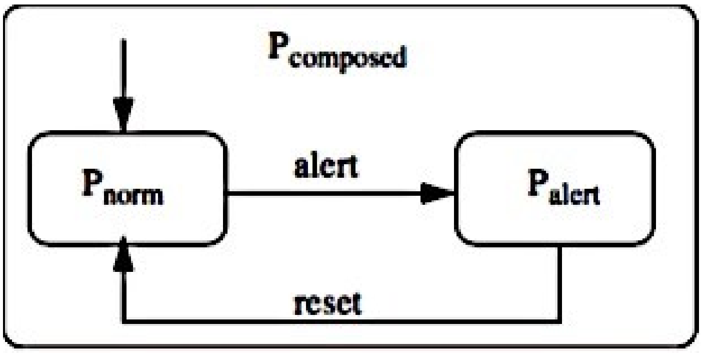


Fig. 4.5 Schematic representation of the sequential composition [54].

One type of policy compositions supported by SANTA is the sequential composition (see Figure 4.5). With this type, it is possible to define how policies evolve over the time and react in response to an occurrence of external events. In various access control scenarios there are frequent situations when security requirements are situation-specific and only apply when certain criteria are met. SANTA provides policy designers to incorporate such situation in their policies in a flexible manner – it is possible to compose policies from individual rules to reflect specific contexts and circumstances, as shown in Table 4.5.

Example 16: Intrusion detection. This example represents a typical situation in the context of an intrusion detection system, which is expected to trigger an intrusion alert and automatically block certain functions of the managed system in response to a detected intrusion. To achieve such functionality, two policies are defined – p_{norm} captures the protection requirements for a normal operational mode, and p_{alert} defines a policy to be enforced in case of detected intrusions (details of these policies are omitted). Accordingly, the policy composition states that the normal operational mode has to turn into the alert mode as soon as an intrusion alert event **event_alert()** is detected. The policy p_{alert} is then applied until the alert is reset (**event_reset()**). The composition then returns to the normal operational mode with the policy p_{norm}.

policy $p_{norm} :: /* \cdots */$ **end**

policy p_{alert} :: / ∗ ··· ∗ / **end**

policy $p_{composed}$ ::

  repeat ((**unless** $event\_alert()$: p_{norm}) ; (**unless** $event\_reset()$: p_{alert}))

end

Example 17: Intrusion detection with an attribute. As opposed to the previous example, where an external event was required to trigger an alert and reset the system operational mode, this example demonstrates how the same behaviour can be achieved using a status attribute **alert_status**. This attribute can be modelled as part of the managed system, where the true value corresponds to an alert, and the false value represents no alert. Now, the policy mechanism does not need to observe external events, but rather react to changes of the **alert_status** attribute.

policy $p_{composed}$ ::

  repeat ((**unless** $alert\_status$: p_{norm}) ; (**aslongas** $alert\_status$: p_{alert}))

end

Example 18: Procurement. This example considers a development company, who wants to outsource certain parts of their product development chain. The procurement process includes four phases – namely, **tender**, **contract**, **development**, and **acceptance**. In the context of a highly competitive business, it is important to ensure fairness among all stake-holders, such that none of them gets an advantage over the others. In particular, it is important that only high-level information about the product is available to potential contractors under the policy $p-tender$ during the initial tender process. Then, at the contracting stage more details are released to the selected contractor under the policy $p-contract$. There are also circumstances, when subsidiaries are allowed to sub-contract and involve third parties in the procurement process – therefore, it is also important to differentiate between immediate subsidiaries and external sub-contractors based on their status. This can be expressed using policies $p-sub$ and $p-ext$, respectively. Eventually, one and the same policy $p-acc$ applies to all contractors at the end of the procurement process.

```
policy $p_{composed}$ ::
    (unless $contractorSelected()$ : $p-tender$);
    (unless $contractSigned()$ : $p-contract$);
    (unless $developmentComplete()$ :
    if $isSubsidiary(Contractor)$
        then $p-sub$
        else $p-ext$);
    $p-acc$
end
```

Table 4.5 SANTA policies and compositions.

4.6.2 Parallel composition: policy union, intersection and difference

The second type of policy compositions supported by SANTA is the parallel composition. With this type, multiple policies can be enforced simultaneously, in parallel to each other. In its simplest form, policies can be combined by merging their rules, as shown in Table 4.6.

Example 19: Merging simple policies (union). This set of examples demonstrates how policies can be composed in a parallel manner. The policy p_{open} is an an open policy, which only defines denials and allows anybody (**S**) to perform the action **a** on the object **o**, by default. Similarly, the policy p_{closed} is a closed policy, which only states permissions and allows the action **a** on the object **o** explicitly. Finally, it is possible to merge the two policies to produce a parallel composition. The lengthy notation can also be shortened, as it contains redundant statements.

policy p_{open} ::

  decide(**S**, **O**, **A**) **when** 0 : **not deny**(**S**, **O**, **A**)

end

policy p_{closed} ::

  allow(**S**, **o**, **a**) **when** *true*

  decide(**S**, **O**, **A**) **when** 0 : **allow**(**S**, **O**, **A**)

end

policy $p_{composed}$ ::

  allow(**S**, **o**, **a**) **when** *true*

  decide(**S**, **O**, **A**) **when** 0 : **allow**(**S**, **O**, **A**)

  decide(**S**, **O**, **A**) **when** 0 : **not deny**(**S**, **O**, **A**)

end

policy $p_{composed}$ ::

  allow(**S**, **o**, **a**) **when** *true*

$\textbf{decide}(\mathbf{S},\mathbf{O},\mathbf{A})$ **when** $0 : \textbf{allow}(\mathbf{S},\mathbf{O},\mathbf{A})$ **or not** $\textbf{deny}(\mathbf{S},\mathbf{O},\mathbf{A})$

end

Example 20: Rule syntax vs semantics. As it follows from the examples above, the intersection of the two simple policies is empty according to their syntax. Nevertheless, the semantic of the two policies are essentially the same – that is, the intersection of the policies is expected to be p_1, which is equivalent to p_2. Therefore, there is a requirement to introduce an operator for policy compositions, which would also capture the semantics of the policies. A potential way of achieving this functionality is to evaluate individual policies constituting the policy composition independently from each other – that is, as a self-contained entity. As a result, the policy composition (i.e., union, intersection, or difference) is defined using a decision rule, which considers outcomes of the both policies (please refer to Example 21).

policy p_1 :: $\textbf{allow}(\mathbf{S},\mathbf{O},\mathbf{A})$**when** *true* **end**

policy p_2 :: $\textbf{allow}(\mathbf{S},\mathbf{O},\mathbf{A})$**when** $0 : true$ **end**

Example 21: Policy composition (intersection). This example represents an intersection of two policies p_1 and p_2. To express this kind of composition, we use the ternary operator par. The policy p_3 in this example is defined by the outcomes of policies p_1 and p_2. It defines the intersection of the two policies as an authorisation decision, which is made only if both policies p_1 and p_2 agree on the outcome. The advantage of this approach is that the meaning of the component policies as a hybrid combination of a positive, a negative and a decision rule is preserved. The **par** operator is able to capture the semantics of the compositional policy, which can then be referred to in the de-confliction policy that defines whether decisions have to be made by the policy composition. The composition preserves the semantics of the component policies. To remove the need to explicitly specify a de-confliction, it is possible to omit the **deconflict** part of the **par** construct. In this case, the example can be shortened to the one presented in Example 22.

policy p_1 :: $\mathbf{allow}(\mathbf{S},\mathbf{O},\mathbf{A})\mathbf{when}\; true$ **end**

policy p_2 :: $\mathbf{allow}(\mathbf{S},\mathbf{O},\mathbf{A})\mathbf{when}\; 0 : true$ **end**

policy p_3 :: p_1 **par**p_2 **deconflict** { $\mathbf{decide}(\mathbf{S},\mathbf{O},\mathbf{A})$ **when**
$p_1.\mathbf{decide}(\mathbf{S},\mathbf{O},\mathbf{A})$ **and** $p_2.\mathbf{decide}(\mathbf{S},\mathbf{O},\mathbf{A})$ }

Example 22: Policy composition. This example illustrates a situation when the policy p_1 will become effective after 10 time units, if the policy p_2 also becomes effective. In this example, the policy p_3 is defined by the outcomes of the policies p_1 and p_2. The policy p_3 defines the intersection of the two policies – an authorisation decision is only made if and only if both policies p_1 and p_2 agree on the outcome.

policy p_1 :: **allow**$(\mathbf{S},\mathbf{O},\mathbf{A})$**when** true **end**

policy p_2 :: **allow**$(\mathbf{S},\mathbf{O},\mathbf{A})$**when** $0:\mathit{true}$ **end**

policy p_3 :: **end**

p_2 **par** $(10:p_3;p_1)$

Table 4.6 SANTA policy union, intersection and difference.

Existing limitation: insufficient support for data privacy In the previous section, we demonstrated the capabilities of the SANTA language to handle various security- and access control-related use case scenarios. As we have seen, using SANTA it is possible to define and enforce policies governing access to bank accounts, electronic submission systems, procurement systems, etc.

Moreover, as we have seen, the application scope of SANTA is not limited by specific platforms – it can apply to relatively small-scale systems and large enterprise concurrent systems with multiple users trying to access resources. Therefore, it is also expected to be applied in the cloud context ‘out of the box’. As demonstrated by the use case examples, all of them can be executed in a wide range of distributed systems, including cloud environments.

Cloud environments are complex virtualised environments, which are characterised with large amounts of data being dynamically transferred in and out. Every single moment, cloud users upload and download gigabytes of their personal data, and cloud-hosted software handles terabytes of potentially sensitive business data. In these circumstances, it is essential to ensure that these amounts of data remain secure and private. Data privacy and protection have been seen among the key challenges for the comprehensive adoption of cloud services.

Admittedly, various techniques, such as data encryption, data replication and recovery, have proven to be useful when data was lost or stolen by an unauthorised/malicious user. However, what if would not escape from the owner’s control in the first place? In other words, it might make more sense to implement an efficient access control mechanism, which would enable data privacy throughout the whole lifecycle of users' data sets within cloud environments. The life cycle of cloud-hosted data typically includes the following phases:

- Data is uploaded by the user over the network
- Data is stored on a cloud server
- Data is moved from one cloud server to another transparently to the user
- Data is downloaded by the user

Arguably, all of these phases are associated with a potential threat of an unauthorised access to data. Accordingly, as far as cloud environments are concerned, it is essential to ensure that data remains private at all times – one of the fundamental requirements for creating a trusted cloud service and attracting customers.

However, as we have demonstrated with the list of use case examples, the SANTA language at its current stage is incapable of addressing the pressing challenge of enabling data privacy across multiple network locations, such as the cloud and the private network. ITL is considerably good at handling the history of data access over a period of time, but currently is not expressive enough to capture the location dimension, but location-aware data privacy is currently beyond existing capabilities of the SANTA language due to the lack of corresponding expressivity power and underpinning logical formalism.

In this light, it becomes important to employ an approach to effectively and efficiently differentiate between various network locations, and – at the same time – combine it with the existing access control policies. Such an approach, as described in the next sections, is based on the logical notion of *location* that lies at the core of topological logic.

4.7 Formalising the notion of location and location transition

In their relevant work [75], Rescher and Garson investigated the notion of place, position, and topology, and possible ways of capturing these concepts with a logical formalism. As a result, the proposed a family of logical systems of so-called positional or topological logic, which extend the traditional temporal logical systems with locative or place logic [96, 84]. In simple words, the proposed topological logic enables differentiation between various positions, be it a physical location (e.g., defined by Cartesian coordinates), or a virtual/network (e.g., defined by its network IP address). When combined to the existing SANTA language and its ability to express access control policies, this key feature of the topological logic has the potential to facilitate the desired functionality of location-awareness and policy transition.

4.7.1 Theoretical underpinnings of Topological Logic

We start explaining the topological logic from an existing system of standard propositional logic (such as, for example, ITL), which is underpinned spatially indefinite propositions – that is, logical propositions are evaluated, regardless of any location or position. The goal is to extend this system with support for the logical notion of location. To do so, we introduce the parameterised operator P_α, where $P_\alpha(p)$ is to be interpreted as 'the proposition p is realised in the location α', where α may be any element of a range of locations. The range of locations, in these circumstances, is a very broad concept, which include any spatial positions defined by any positional scheme (e.g., Cartesian coordinates, geographical coordinates, or network IP addresses). For example, speaking in terms of the previously presented use case scenario, the set of locations may consist of three different types, as summarised in Table 6.3.

Having defined the parameter α as a place, location, position, and other similar concepts, the considered proposition P_α should be interpreted as a propositional function of this parameter type. In other words, if α represents a location, then p is a spatially indefinite proposition that can be broadly read as “something is happening in the location α”. Accordingly, the following three fundamental axioms of the topological logic can be stated [75]:

$$P_\alpha(\sim p) \equiv \sim P_\alpha(p) \tag{4.1}$$

$$P_\alpha(p \;\&\; q) \equiv [P_\alpha(p) \;\&\; P_\alpha(q)] \tag{4.2}$$

$$(\forall\alpha)(P_\beta[P_\alpha(p)]) \equiv P_\Box[(\forall\alpha)P_\alpha(p)] \tag{4.3}$$

Let us consider these two axioms one by one. Axiom 4.1 states that if not-p is true in some location α, then it is not possible for p to be true in that location, and vice versa. In simpler terms, the axiom suggests that a logical proposition p can only be either true or false in the location α, whereas any other values, such as ‘undefined’, ‘infinite’, ‘inapplicable’, etc. are not allowed. From this perspective, it is similar to the binary Boolean logic.[2] From an access control point of view, such a location-aware access control policy is dependent on a location, and, when evaluated, is expected to return either true or false.

Axiom 4.2 asserts that if a conjunction of two different propositions p and q is true in some location α, then each of the conjuncts is also true at that specific location, and vice versa. In access control terms, the axiom implies that if two policies hold in a specific location, then each of them independently also holds in that location.

Next, it is also necessary to enable quantification of the parameter α in P_α, so as to enable the sufficient power of the described topological logic. So far, for Axioms 4.1 and axiom2, we supposed an initial universal quantifier with respect to α. In a similar manner, we suppose that every asserted proposition is to be asserted universally with respect to its (otherwise

[2] It is worth noting that if the latter condition was dropped and a third truth-value would be allowed, the equivalence connective of Axiom 4.1 would need to be replaced with an implication.

unqualified) parameters. Consequently, given the usual machinery of quantificational logic, the Axiom 4.3 can be derived.

Given that parameter values are natural numbers, Axiom 4.3 can be further extended for universal quantification as a potentially infinite conjunction of individual propositions, the left-hand side of Axiom 4.3 can be expressed as:

$$P_\beta(P_0(p)) \ \& \ P_\beta(P_1(p)) \ \& \ P_\beta(P_2(p)) \ \& \ ...$$

Similarly, its right-hand can be represented as:

$$P_\beta[P_0(p) \ \& \ P_1(p) \ \& \ P_2(p) \ \& \ ...]$$

According to Axiom 4.2, these two extended parts are essentially equivalent.

Taken together, the presented three main Axioms 4.1, 4.2, and 4.3 underpin the topological logic and will be the main reference point for implementing the location-aware functionality in the context of the proposed approach.

4.7.2 Combining ITL/SANTA with Topological logic

By offering the notion of location α, the topological logic provides a way of combining multiple propositions p (e.g., expressed in ITL) into a compositional set of policies, where some propositions hold in one location, whereas some other propositions hold in another location. In this context, a proposition p is represented by a SANTA access control policy, whereas the set of possible locations α's may represent network locations.

More specifically, by defining the exhaustive list of network locations as α's it becomes possible to differentiate between various locations (i.e., personal computer, enterprise domain, or public network), and apply corresponding access control policies depending on a specific location. This combination of ITL with Topological Logic can be generally represented as $P_A(p)$, where the p is a proposition in ITL representing a SANTA policy (such as the access control policy examples in Section 4.6), and P_A is a proposition in Topological Logic representing physical and logical locations, in which these policies will apply. The three fundamental Axioms 4.1, 4.2, and 4.3 of Topological Logic will also apply. For example, taking Example 19 as an illustrative case, we show below how it can be enhanced with support for location-awareness.

Example 19 (Re-visited): Simple access control policies with location-awareness. The original set of examples demonstrated how two simple policies p_{open} and p_{closed} can be composed in a parallel manner. The enhanced policies $P_A(p_{open})$ and $P_B(p_{closed})$ also include topological propositions, which state that the policies should be evaluated positively and actions **A** on the object **O** are allowed, only if subject **S** is in locations A and B, respectively. Similar to the original example, the two policies can be merged into $P_{A,B}(p_{composed})$, which combines the two location-aware policies together.

policy $P_A(p_{open})$ ::
decide$(\mathbf{S},\mathbf{O},\mathbf{A})$ **where S in A and when** 0 : **not deny**$(\mathbf{S},\mathbf{O},\mathbf{A})$
end
policy $P_B(p_{closed})$ ::
allow$(\mathbf{S},\mathbf{O},\mathbf{A})$ **when** $true$
decide$(\mathbf{S},\mathbf{O},\mathbf{A})$ **where S in B and when** 0 : **allow**$(\mathbf{S},\mathbf{O},\mathbf{A})$
end
policy $P_{A,B}(p_{composed})$ ::
allow$(\mathbf{S},\mathbf{O},\mathbf{A})$ **when** $true$
decide$(\mathbf{S},\mathbf{O},\mathbf{A})$ **where S in A and when** 0 : **allow**$(\mathbf{S},\mathbf{O},\mathbf{A})$
decide$(\mathbf{S},\mathbf{O},\mathbf{A})$ **where S in B and when** 0 : **not deny**$(\mathbf{S},\mathbf{O},\mathbf{A})$
end

4.8 Summary

In this chapter, we familiarised the reader with the SANTA policy language and potential application scenarios, in which it can be used. One of the main advantages of the language is its support for the temporal and sequential constraints – that is, it is possible to define how various subjects request access to objects one after another, or when the history of previous access actions is taken into consideration. These application scenarios, however, mainly concern the 'traditional' notion of access control and do not consider the dynamic nature of cloud environments, where data and users might migrate from one security domain to another, and, therefore, access control requirements might change accordingly with respect to the changing physical or logical locations. As a way of supporting such scenarios, the chapter introduced Topological Logic and demonstrated how it can be integrated with SANTA. In the next two chapters, we explain how this issue can be potentially addressed by our proposed approach by first presenting a high-level design of the framework and then proceeding with implementation details.

Chapter 5

Access Control Policy Framework Design

Objectives:

- To provide a general overview of the proposed policy framework.
- To describe the architecture of the policy framework.
- To explain how the policy framework components interact.
- To describe the concepts of location, location-awareness, and policy transition.

5.1 Introduction

A security policy is the basis of an organisation's information security. Many organisations have information security and access control policies in place to ensure that their information is always secure and remains protected. However, having a security policy document in itself is not enough. It is very important to ensure that the contents are actually implemented and enforced to achieve truly effective access control and security. Given the rapidly increasing amounts of data to be analysed and policies to be evaluated, this process needs to be automated as much as possible. This means that there has to be a dedicated software, which would handle the access control tasks in a timely, reliable and efficient manner.

A common and established practice to implement such an access control software is to rely on a modular architecture and implement several loosely-coupled components. Accordingly, control architectures often distinguish policy enforcement points and policy decision points. Policy enforcement points intercept access to protected application resources and request authorisation decisions from a policy decision point. A policy decision point evaluates authorisation decision requests relative to a security context and returns the evaluation result to the policy enforcement point. If the evaluation result indicates sufficient privileges the policy enforcement point allows the initial requester to access the protected resource, otherwise access is blocked.

This section will further extend this preliminary glimpse on the proposed access control framework. The section is dedicated to the description of the high-level architecture, which is designed in a modular manner, and each element's role and functionality are explained in details with respect to a sample use case scenario, thus demonstrating the existing challenges and how they can be addressed by the proposed solution.

5.2 Sample use case scenario: a file storage service

Before proceeding with an actual explanation and discussion of the proposed architecture, it is worth presenting a sample use case scenario, which will serve to explain the architecture. Please note that the scenario is intended to better reflect and highlight some features of the proposed solution, and therefore is correspondingly simplified.

The use case scenario focuses on heterogeneous (hybrid) clouds – complex environments, where data may be transferred between various administrative domains and network locations. Therefore, they require advanced access control mechanisms to be put in place. Main aspects making traditional security mechanisms less applicable to such cloud scenarios are the following [25]:

- *Nature of the cloud*: because of the cloud's intrinsic characteristics, such as dynamic scalability, virtualisation and service abstraction, and geophysical location transparency, hosted software systems have no fixed underlying infrastructure (due to frequent virtual machine migrations) and security boundaries. In these circumstances, it becomes difficult to identify and isolate a single physical resource, which was compromised or put the system at risk in any other way. In the case of hybrid clouds, these issues are taken to the next level of complexity, as there are more than one cloud party involved in such a scenario.

- *Multiple parties with no agreed security policy*: as suggested by the service-oriented nature of cloud computing, various levels of a cloud offering may belong to different service providers (e.g., many PaaS providers use Amazon Web Services' infrastructure as a service). In these circumstances, there might be a conflict of interests between various stakeholders, as there is no unified security framework applicable to all interested parties.

- *Mutliple users and a single physical host*: the (hybrid) cloud relies on the virtualisation technology, enabling multiple tenants to share one and the same physical space. This opens new opportunities for unauthorised access to private resources.

- *The scale and complexity of cloud-based systems*: in the age of Big Data, avalanches of data are being generated, processed, and stored in the cloud [35, 31], which also means that corresponding security mechanisms need to cope with similar amounts of data to maintain a stable level of security.

A representative example, illustrating the listed challenges, are common file sharing cloud services, such as Dropbox.[1] Dropbox is a file hosting service that offers cloud storage, file synchronization, personal cloud, and client software.

Dropbox creates a special folder on the user's computer, the contents of which are then synchronised to Dropbox's servers and to other computers and devices that the user has installed Dropbox on, keeping the same files up-to-date on all devices. Dropbox uses a freemium business model, where users are offered a free account with a set storage size, with paid subscriptions available that offer more capacity and additional features. Dropbox offers computer apps for Microsoft Windows, Apple MacOS, and Linux computers, as well as mobile apps for iOS, Android, and Windows Phone smartphones and tablets.

The Dropbox software enables users to drop any file into a designated folder. The file is then automatically uploaded to Dropbox's cloud-based service and made available to any other of the user's computers and devices that also have the Dropbox software installed, keeping the file up-to-date on all systems. When a file in a user's Dropbox folder is changed, Dropbox only uploads the pieces of the file that have been changed, whenever possible. When a file or folder is deleted, users can recover it within 30 days. Dropbox also offers synchronisation support over a LAN, where, instead of receiving information and data from the Dropbox cloud servers, computers on the local network can exchange files directly between each other, potentially significantly improving synchronisation speeds. Dropbox originally used Amazon's S3 storage system to store user files, but between 2014 and 2016 they gradually moved away from Amazon to use their own hardware, referred to as 'Magic Pocket'.[2] Dropbox uses SSL transfers for synchronisation and stores the data via AES-256 encryption. The functionality of Dropbox can be integrated into third-party applications through an application programming interface (API). Dropbox prevents sharing of copyrighted data, by checking the hash of files shared in public folders or between users

[1] https://www.dropbox.com/

[2] https://blogs.dropbox.com/tech/2016/05/inside-the-magic-pocket/

against a blacklist of copyrighted material. This only applies to files or folders shared with other users or publicly, and not to files kept in an individual's Dropbox folder that are not shared.

It is worth noting that Dropbox has already been criticised for its insufficient security and data privacy facilities. For example, in June 2011, an authentication problem let accounts be accessed for several hours without passwords. In June 2013, there was a leak of multiple government documents with information that Dropbox was being considered for inclusion in the National Security Agency's PRISM surveillance program. Next, 68 million Dropbox account passwords leaked on the Internet in August 2016. These are just a few examples of the issues Dropbox have been facing recently, and the actual list of incidents might be somewhat longer.

As it follows from this description, the Dropbox service is designed to operate in a mixed environment, which includes a public cloud (for global access and file sharing), private local network (for enterprise access and file sharing), and local folder on an end device (for local storage and access). Accordingly, a resource can possibly be located in any of the three locations (or replicated across all of them), each of which is characterised with a different set of access control requirements. This way, whenever a user requests an access to a Dropbox file, there has to be an evaluation procedure, which will take into account such metadata as the user's attributes (e.g., location and access rights), the resource's attributes (e.g., location and sharing permissions), and access control policies currently in place.

Taken together, this consideration outlines the envisioned access control mechanism, which is supposed to intercept incoming access requests and be able to differentiate between various physical and logical locations so as to apply different policies based on that. Accordingly, in the next sections of this chapter we first explain the general conceptual architecture of the proposed solution, and then extend it with the notions of location and policy transitions – key concepts, underpinning the whole proposed research.

5.3 Architecture of the access control policy framework

As was previously discussed, the topic of access control is not novel and has been attracting researchers' attention for several decades. Accordingly, in our work we aimed to build upon this existing work so as not to 're-invent the wheel' wherever possible. Accordingly, the main point of reference in the context of this research was the widely adopted Attribute-Based Access Control, and – more specifically – the software reference model, suggested by the National Institute of Standards and Technology (USA) [50].

As we have already discussed throughout this document, a security (i.e., access control) policy is the fundamental building block of an organisation's IT security. Many companies have information security policies in place to ensure that their sensitive business information remains secure and protected at all times. Admittedly, simply defining security policies and storing them as a set of plain documents, however, is not enough to maintain a stable and high level of information protection and security. It is very important to ensure that the contents of this document are actually implemented to be effective. In other words, there has to be a mechanism, which would – on one hand – enforce these security (i.e., evaluate incoming data against this predefined set of policies), and – on the other – monitor the system context to ensure that policies to be enforced are indeed valid and match the current situation. From this perspective, this organisation is akin to the three branches of government, as adopted in many countries. More specifically, the legislative power is represented by the set of security/access control policies, whereas the executive and the judicial powers are represented by a corresponding enforcement mechanism, responsible for putting the policies in action and evaluating them as required.

Accordingly, the proposed approach to implementing access control in hybrid cloud environments is also based on a modular architecture. Firstly, it is composed of a declaratively-defined access-control policy base. It also includes an enforcement mechanism – a software component responsible for evaluating the policy base against the current situation. The enforcement mechanism, in its turn, is itself a modular component, which is composed of several other conceptual elements, as it will be further explained below.

5.4 Conceptual architecture of the proposed access control policy framework

Having introduced the target use case scenario, we now proceed with an explanation of the conceptual architecture of the policy framework. For demonstration purposes, this explanation will specifically focus on the use case scenario at hand – in practice, the range of application scenarios is expected to be much wider. Schematically, the architecture is depicted in the diagram in Figure 5.1.

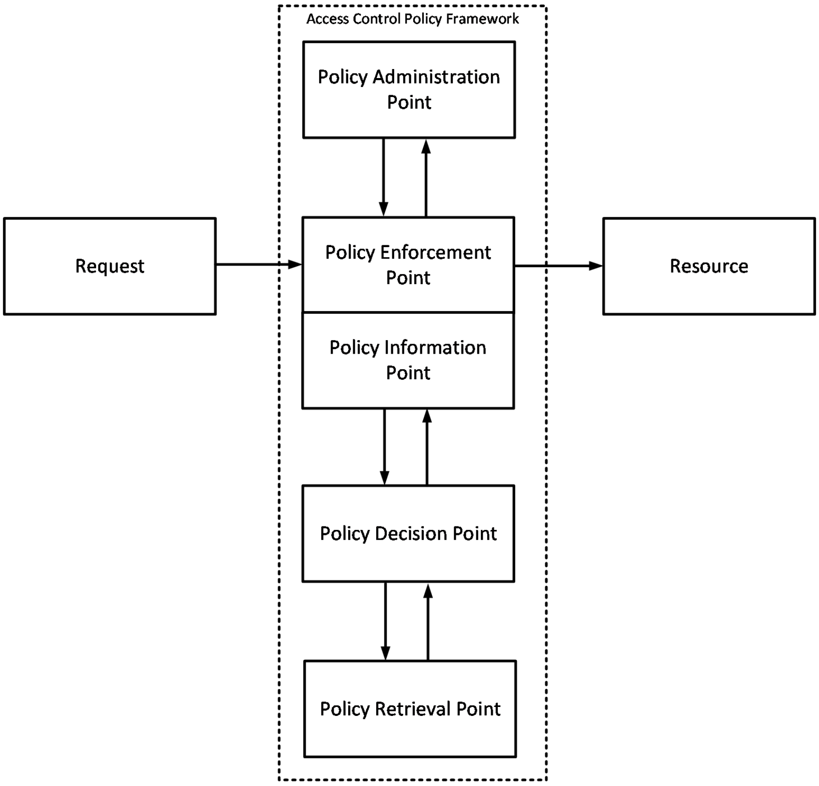


Fig. 5.1 A high-level architecture of the proposed access control policy framework.

The proposed access control framework (depicted as the central element in Figure 5.1) consists of several main elements. These elements are explained in more details below.

5.4.1 Policy Administration Point

Policy Administration Point is a software component that is responsible for managing policies. Typically, it is represented by a user interface for creating, managing, testing, and debugging access control policies, and storing them in the appropriate repository – i.e., a Policy Retrieval Point. The main task of a PAP is to provide support for editing, testing and evaluating policies to ensure they meet the intended requirements.

5.4.2 Policy Enforcement Point

A Policy Enforcement Point (PEP) enforces policy decisions in response to a request from a subject requesting access to a protected object. It is responsible for intercepting incoming access requests to a specific cloud-based resource (e.g., a document, a file, a database, etc.), as well as gathering information about the current context, such as the resource location, the location from where the request was generated, user credentials, current time, etc. The PEP is expected to be deployed on a network device (e.g., a gateway, a router), on which policy decisions are carried out or enforced. From this perspective, it can be seen as a component that serves as the gatekeeper and the 'front door' to a digital cloud-based resource. When a user tries to access a file or other resource on a computer network or server, the PEP will describe the user's attributes to the Policy Information Point (explained below), and transfer the request to a Policy Decision Point (explained below), which is actually responsible for taking an access control decision. The PEP is usually specific to an application and cannot be re-used for different applications. Briefly, the functionality of the PEP can be summarised as follows:

1. The PEP receives the incoming access request.

2. The PEP extracts subject and object attributes.
3. The PEP uploads these attributes to a Policy Information Point.
4. The PEP passes the request to a Policy Decision Point.
5. The PEP receives a decision response from the Policy Decision Point.
6. The PEP enforces the decision by either permitting access or denying access to the request.

5.4.3 Policy Information Point

To compute access decisions, the Policy Decision Point must have information about the attributes of both subject and object. Simply put, the system needs to know who exactly is trying to access a resource, and what kind of resource it is. This information is provided by a Policy Information Point (PIP), which is a software component that serves to store attributes received from the PEP and to retrieve these attributes when required by the Policy Decision Point. This information is required during the policy evaluation process to make a policy evaluation decision.

5.4.4 Policy Decision Point

A Policy Decision Point (PDP) is a software component, responsible for the actual evaluation of policies, it makes authorisation decisions for itself or for other system entities that request such decisions. It computes access decisions by evaluating the applicable access control policies with respect to the subject and object attributes, which are retrieved from the Policy Information Point. In other words, the PDP makes the determination of whether or not to authorise an incoming request, based on available information (i.e., attributes) and applicable security policies.

5.4.5 Policy Retrieval Point

This is the actual location, where access control policies are stored. It is important to make this location both easily-accessibly and well-protected, as it contains the crucial knowledge, underpinning the security of the whole system. Typically, access authorisation policies are stored as records in a relational database, or semi-structured documents in the file system.

5.5 Main Benefits and Features

It is worth noting that the PAP, PEP, PIP, PDP, and PRP functionality can be either distributed or centralised, and may be physically and logically separated from each other. For example, an enterprise company could establish a centrally-controlled enterprise decision service that evaluates attributes and policy, and issues decisions that are then passed to the PEP. This way, all the points are deployed on a single machine. This allows for central management and control of subject attributes and policies. Alternatively, local organisations within the enterprise may implement separate physical or logical locations, which are used to deploy individual elements of the access control system.

The following potential benefits of the proposed architecture can be identified:

- *A Policy Enforcement Point is a single point of access*: this means that all incoming requests will pass through this component, and not a single request will pass unnoticed. This way, an increased level of system security and access control is achieved. Since any request is intercepted by the PEP, an unauthorised request is never expected to get to an application, service or data. Therefore, it is much harder to compromise the application. The same is not true if the PEP and PDP are essentially implemented in the application's security model.

- *Policies are decoupled from applications and services*: defined in a declarative manner, policies are stored in the PRP and can be managed independently of applications and services, which can therefore concentrate on providing business value. Moreover, adding new, modifying existing, or deleting old policies is simplified: these activities can be done, so that changes are written to the corresponding component (i.e., the PRP) in a seamless and transparent manner via the PAP.

- *The standardised modular architecture also assumes that some of the elements can be re-used*: this means that already-existing, optimised and reliable solutions can be applied to implement particular functions. For example, a common approach to implement the PRP is to employ a relational database. Since there are no strict requirements, a MySQL database, for example, can be replaced by an Oracle database. Moreover, having this kind of strict separation of concerns between individual elements improves the overall structure of the target system. Auditing, logging or debugging of individual elements is expected to be much simpler than performing these activities across many disparate applications and services. In other words, by being aware of what functionality is implemented by a specific point, it is much easier to track down potential issues and exceptions.

5.6 Sample policy enforcement workflow

Having explained the proposed architecture, we now proceed with an explanation of how the Dropbox use case scenario is expected to be handled by the presented solution.

1. The security administrators, by interacting with the PAP, are able to define (create/-modify/delete) access control policies applicable to Dropbox documents when they are stored on a local machine, on the enterprise network, or on the public cloud server. The policies are then stored in the PRP for later use.

2. The client sends a request to access a protected application resource (i.e., a Dropbox-hosted file).

3. The PEP intercepts the request to the resource.
4. The PEP extracts relevant attributes from the request (the user's IP address, location, user credentials, etc.) and writes them to the PIP.
5. The PEP retrieves relevant attributes of the requested resource (the resource's IP address, location, access rights and sharing permissions, etc.) and writes them to the PIP. The location can be either the local enterprise network or the public cloud.
6. The PEP routes the incoming request to the PDP for evaluation.
7. The PDP receives the incoming request, retrieves relevant attributes for the corresponding user and resource, as well as the access control policies associated with the resources from the PRP.
8. Taking into account the user and resource attributes (which represent the security context in this case), the PDP evaluates the policies and generates an authorisation decision, which is either positive or negative.
9. The PDP sends the authorisation decision to the PEP.

Based on the authorisation decision, the PEP decides whether to grant the incoming request with access to the resource or not. By default, if the response is positive, the PEP filters passes, and the original client request for the resource is authorised, and the policy flow continues on the success path.

5.7 Location and Location-awareness

The core element of the proposed access control framework is the set of access control policies, which are defined by security administrators and stored in the PRP. The policies are expected to be defined using the SANTA policy language – a declarative language, such that potential modifications of the policy base would take place in a transparent and seamless manner.

However, as it was previously demonstrated by the use case scenario, heterogeneous cloud systems are complex virtualised environments, characterised by large amounts of data being dynamically transferred in and out. Every single moment, cloud users upload and download gigabytes of their personal data, and cloud-hosted software handles terabytes of potentially sensitive business data. In these circumstances, it is essential to ensure that these amounts of data remain secure and private in all locations and at all times. This means that the SANTA language needs to be extended with the notion of location, as explained in the previous Chapter and further detailed below.

"Location-awareness, the ability to determine geographical position, is an emerging technology with both significant benefits and important privacy implications for users of mobile devices such as cell phones and PDAs" [63]. Location can be determined either *i)* internally by devices, or *ii)* externally by systems and networks with which devices interact. The former case is underpinned by the advances in the embedded technologies and microelectronics, which enabled equipping portable mobile devices with the Global Positioning System (GPS). This way, portable devices are aware of their precise geo-physical location, and, as a result, the current country and region. The latter case is realised through the WHOIS protocol and querying a corresponding online service [40]. Each of the two ways (or a combination of both) provides relatively precise information on the current location – both physical (i.e., country and region) and logical (i.e., networks and subnets) – of a device, thus paving the way for using this information in a wide range of applications and services. Examples of such applications and services include navigation systems, social networks, file hosting systems, online games, etc.

The proposed notion of 'location' is illustrated by Figure 5.2, which extends the traditional established concepts of the access control model (i.e., subject, object, action, and purpose).

According to this view, the policy enforcement mechanism, when evaluating a policy, needs to take into consideration not only who requests access to data, but also where he/she and the requested data are currently located.

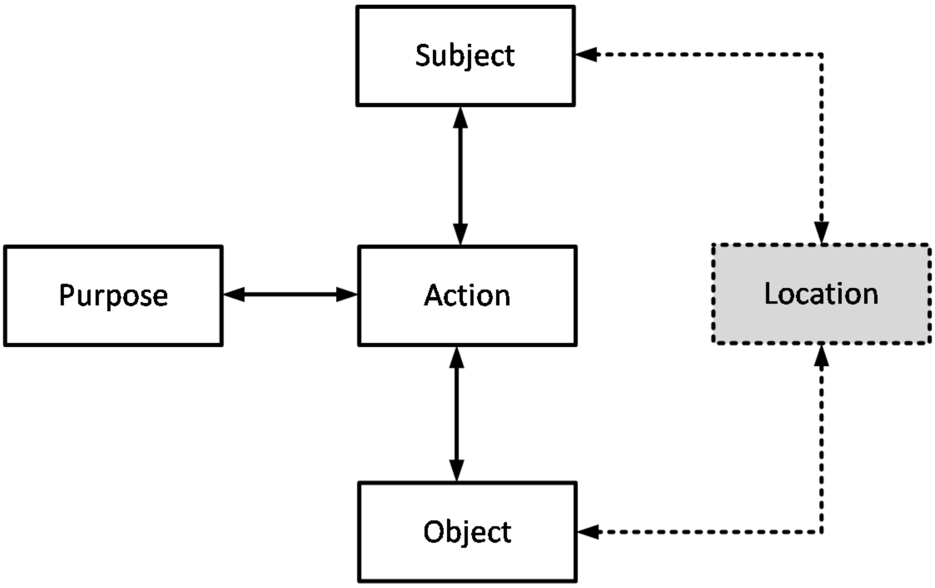


Fig. 5.2 Conceptual architecture of an access control policy, enhanced with the notion of location.

5.8 Policy Transition

Understanding the concept of location is not yet enough to fully enable the differentiated (i.e., location-aware) access control. The access control mechanism also needs to detect situations, when policy subjects or objects change their locations, thus requiring the evaluation mechanism to act accordingly. This presented concept of 'policy transition' (i.e., change of access control policies from one to another, based on the current location of the subject or the object) is a key concept of the proposed solution.

One potential method of implementing the location-based policy transition is through compositional policies [55, 68]. A compositional policy is a policy, which allows to handle sequential and parallel composition of atomic rules. These rich expressivity and flexibility, however, come at a cost – there is a pressing requirement of handling and resolving potential conflicts when two or more simple rules are combined into a compositional policy. In any case, a compositional policy can include several ‘atomic’ rules, each of which concerns only a single aspect (i.e., data location) – for example, a policy may contain several rules, which specify how to handle data access when information is *i)* in the private network, *ii)* transferred over the public network, *iii)* is in the public cloud. From the policy enforcement point of view, however, the challenge is to decide when to change from one individual policy to another so as to meet the current location’s requirements. To achieve this goal, it is expected that the proposed system is able to continuously monitor network locations of all the involved parties, and apply corresponding access control policies for different network locations, as well as to be able to switch from one policy to another following a change in the monitored network locations.

5.9 Summary

Access control policies are essential plans of action, which enable security administrators to control and evaluate who can access information, how long to retain information and how effectively individuals are complying with the policies themselves. In complex heterogeneous environments, it is often required to enable access control based and enforce corresponding policies based on the current location (both physical and logical) of the user and the requested resource. Accordingly, in this chapter we provided an overview of the conceptual architecture of the future access control software framework. The proposed design is based on the NIST’s model of attribute-based access control and partially re-uses its reference architecture for implementing automated access control in distributed network environments. Among the main benefits of the proposed architecture we can distinguish its modularity, increased security due to a single-point access, and a declarative approach to defining the policies. The latter is achieved by means of an existing policy language SANTA, which is further extended with the notions of location, location-awareness, and policy transition.

Chapter 6

Proof of Concept Through a Case Study

Objectives:

- To prove the viability of the chosen approach.
- To present a relevant case study focusing on access control requirements and highlighting the need for location-aware policy enforcement.
- To demonstrate how the proposed approach can address the outlined use case scenario requirements.
- To present the design and implementation details of the proposed system.

6.1 Introduction

This chapter is intended to provide the reader with a more in-depth understanding of the proposed approach through a case study. The case study is based on a real-world use case scenario, which – on the one hand – fully demonstrates the potential of the presented approach, and – on the other – is tailored for clarity, making it easy to understand. Accordingly, the chapter first introduces the reader to the use case scenario, highlighting the main relevant aspects of it. Next, it summarises access control requirements, associated with the use case scenario focusing on location transition, and outlines – in a high-level manner – desired features for an access control policy enforcement. Finally, these requirements and features are captured in a more formal manner using a logical formalism to enable location-aware policy enforcement.

6.2 Use case scenario: a corporate cloud storage service

Taking the Dropbox scenario presented in the previous chapter as a starting point, we now present and explain a case study that will serve to prove the viability of the overall approach through multiple access control requirements to be addressed. Please note that the scenario is intended to better reflect and highlight some features of the proposed solution, and therefore is correspondingly streamlined.

The target scenario focuses on a corporate file sharing system adopted by a large enterprise company. Employees of the company are expected to use the file sharing system to store and share work-related resources through a corporate network. Accordingly, there are three main file storage locations involved:

- *Local storage*: files are stored locally on an employee's personal computer or a personal virtual drive. In both cases, only the creator/owner of a file is expected to manipulate and manage this file. Typically, files remain in a local storage upon creation, and are supposed to be shared to the public storage at some point (provided they are work-related and are not intended for personal use only). Running a local storage on an end device, such as a personal computer, a tablet, or a smartphone, requires installing a client application that takes care of the network communication, user authentication, sharing, etc. in a manner similar to the existing file sharing systems, such as Dropbox, Google Drive, or Microsoft One Drive. From a software version control perspective, the local storage can be thought of as a local repository, where software developers can freely experiment with various code modifications before merging the tested updates to the public repository.

- *Internal corporate storage*: at some point, system users may decide to share their files with the rest of their colleagues by uploading files to a private network-accessible location within the enterprise network domain. Such a common location is expected to be deployed on a considerably powerful server machine, equipped with sufficient storage and networking capabilities so as to handle large amounts of information being accessed simultaneously from various locations. Similar to local machines, the server machine is also running the file sharing software responsible for file synchronisation, user authentication, networking, etc. Further drawing parallels with software version control system, the central server machine can be seen as a central repository, where source code updates are committed for sharing with the rest of collaborating peer developers.

- *External corporate storage*: whenever the capacities of the internal storage server are exhausted, the file sharing system may extend its space by pushing information to an external public cloud, following the principles of the so-called 'cloud bursting', thereby creating a common virtual space, transparent to the user. The deployed file sharing system also allows accessing the internal repository and personal machines from outside the network enterprise, e.g., for employers working from home or travelling on business trips. Using personal credentials, users can remotely access either the central repository or their personal machines.

The use case scenario deals with a heterogeneous (hybrid) environment, where personal information and files may be transferred between various administrative domains and network locations. Therefore, they require advanced access control mechanisms to be put in place.

The system, however, is equipped with a rather simplistic role-based access control functionality, which only takes into consideration the credentials of a user and corresponding access rights to determine if he/she is able to access and manipulate a file. That is, the existing access control does not consider the actual location of the requested resources (i.e., objects) and the users requesting access to those resources (i.e., subjects). Therefore, this approach is not flexible enough to apply differentiated, location-specific access control policies, which would enable a higher level of protection whenever a 'riskier' and more sensitive location is involved – on the one hand, and better resource utilisation by not applying unnecessarily complex policies when the risk is minimal – on the other.

Accordingly, to further explain and demonstrate how this kind of location-aware access control functionality can be implemented to complement the existing system, the presented scenario is segmented into atomic access control requirements.

6.3 Access control requirements

As it follows from the description of the use case scenario, the enterprise file sharing service is designed to operate in a mixed environment, which includes:

- A public cloud (for global access and file sharing).
- A private local network (for enterprise access and file sharing).
- A local folder on an end device (for local storage and access).

Accordingly, a resource can possibly be located in any of the three locations (or replicated across all of them), each of which is characterised with a different set of access control

Table 6.1 Access control actions and requirements.

Action	Description
ReadObject	An access control subject is allowed to access a remote resource in a read-only mode, with possibility of modifying it.
WriteObject	An access control subject is allowed not only to read a resource, but additionally modify it (e.g., edit, delete, rename, etc.).
CreateObject	An access control subject is allowed to create and upload new resources.

requirements. This way, whenever a user requests an access to a Dropbox file, there has to be an evaluation procedure, which will take into account such metadata as the user's attributes (e.g., location and access rights), the resource's attributes (e.g., location and sharing permissions), and access control policies currently in place.

Taken together, these considerations outline the envisioned access control mechanism, which is supposed to intercept incoming access requests and be able to differentiate between various physical and logical locations so as to apply different policies based on that. Accordingly, in the next sections of this chapter we first explain the general conceptual architecture of the proposed solution, and then extend it with the notions of location and policy transitions – key concepts, underpinning the whole proposed research.

More formally, these main system requirements in the context of the presented use case outline the main principles for a corresponding access control policy to be put in place, as summarised in Table 6.1.

Next, it is also important to differentiate between various types of subjects accessing remote resources, as summarised in Table tab:subjects

Most importantly, it is crucial to differentiate between three types of network location, involved in the current scenario, as summarise din Table 6.3.

Table 6.2 Access control subjects.

Subject	Description
Resource Owner (**RO**)	This is a user who owns a specific resource and originally creates it in his/her own personal working space, after which the resource is uploaded to the private corporate network and a public cloud.
Internal User (**IU**)	This is a user possessing necessary credentials to access common shared resources both from within and outside the organisation and its network.
External User (**EU**)	This is a user who does not belong to the organisation (i.e., does not have official credentials, such as email and password) and is trying to access the resource from outside the organisation and its network domain.

Table 6.3 Access control locations.

Location	Description
Local Machine (**LM**)	This is the local machine of the resource owner, where files are initially created to be further uploaded to the private corporate network and a public cloud.
Internal Network (**IN**)	This is the trusted private network domain, where company employees can share their resources.
External Network (**EN**)	This is any other network location – i.e., external to the enterprise network – from where users can access shared resources.

Table 6.4 ReadObject access control matrix and location-aware access control policy definition.

<table>
<tr><th></th><th>Local Machine (<b>LM</b>)</th><th>Internal Network (<b>IN</b>)</th><th>External Network (<b>EN</b>)</th></tr>
<tr><td>Resource Owner (<b>RO</b>)</td><td>Allow</td><td>Allow</td><td>Allow</td></tr>
<tr><td colspan="4"><b>policy</b> $P_{LM,IN,EN}(p_{ReadObject})$ ::
$\mathbf{decide}(\mathbf{RO},\mathbf{O},\mathbf{A})$ <b>where RO in LM or RO in IN or RO in EN</b> : $\mathbf{allow}(\mathbf{RO},\mathbf{O},\mathbf{A})$
<b>end</b></td></tr>
<tr><td>Internal User (<b>IU</b>)</td><td>Deny</td><td>Allow</td><td>Allow</td></tr>
<tr><td colspan="4"><b>policy</b> $P_{LM,IN,EN}(p_{ReadObject})$ ::
$\mathbf{decide}(\mathbf{IU},\mathbf{O},\mathbf{A})$ <b>where IU in IN or IU in EN</b> : $\mathbf{allow}(\mathbf{IU},\mathbf{O},\mathbf{A})$
$\mathbf{decide}(\mathbf{IU},\mathbf{O},\mathbf{A})$ <b>where IU in LM</b> : $\mathbf{deny}(\mathbf{IU},\mathbf{O},\mathbf{A})$
<b>end</b></td></tr>
<tr><td>External User (<b>EU</b>)</td><td>Deny</td><td>Deny</td><td>Deny</td></tr>
<tr><td colspan="4"><b>policy</b> $P_{LM,IN,EN}(p_{ReadObject})$ ::
$\mathbf{decide}(\mathbf{EU},\mathbf{O},\mathbf{A})$ <b>where EU in LM or EU in IN or EU in EN</b> : $\mathbf{deny}(\mathbf{EU},\mathbf{O},\mathbf{A})$
<b>end</b></td></tr>
</table>

These three types of parameters constitute a matrix of corresponding access rights that define whether a specific user can perform a specific operation on a specific resource from a specific network location, as summarised in Tables 6.4, 6.5 and 6.6.

Using these matrices, it is becoming possible to distil the following main access control principles that will be addressed in the presented use case:

- *Principle 1*: when a resource is still located on the owner's personal machine, no one but the owner himself can only access it for read, write and create operations.
- *Principle 2*: the resource owner has all rights to read and write a resource from any network location.

Table 6.5 WriteObject access control matrix and location-aware access control policy definition.

<table>
<tr><th></th><th>Local Machine (<b>LM</b>)</th><th>Internal Network (<b>IN</b>)</th><th>External Network (<b>EN</b>)</th></tr>
<tr><td>Resource Owner (<b>RO</b>)</td><td>Allow</td><td>Allow</td><td>Allow</td></tr>
<tr><td colspan="4">$\mathbf{policy}\ P_{LM,IN,EN}(p_{WriteObject}) ::$
$\quad\mathbf{decide}(\mathbf{RO},\mathbf{O},\mathbf{A})\ \mathbf{where\ RO\ in\ LM\ or\ RO\ in\ IN\ or\ RO\ in\ EN} : \mathbf{allow}(\mathbf{RO},\mathbf{O},\mathbf{A})$
$\mathbf{end}$</td></tr>
<tr><td>Internal User (<b>IU</b>)</td><td>Deny</td><td>Allow</td><td>Deny</td></tr>
<tr><td colspan="4">$\mathbf{policy}\ P_{LM,IN,EN}(p_{WriteObject}) ::$
$\quad\mathbf{decide}(\mathbf{IU},\mathbf{O},\mathbf{A})\ \mathbf{where\ IU\ in\ IN} : \mathbf{allow}(\mathbf{IU},\mathbf{O},\mathbf{A})$
$\quad\mathbf{decide}(\mathbf{IU},\mathbf{O},\mathbf{A})\ \mathbf{where\ IU\ in\ LM\ or\ IU\ in\ EN} : \mathbf{deny}(\mathbf{IU},\mathbf{O},\mathbf{A})$
$\mathbf{end}$</td></tr>
<tr><td>External User (<b>EU</b>)</td><td>Deny</td><td>Deny</td><td>Deny</td></tr>
<tr><td colspan="4">$\mathbf{policy}\ P_{LM,IN,EN}(p_{WriteObject}) ::$
$\quad\mathbf{decide}(\mathbf{EU},\mathbf{O},\mathbf{A})\ \mathbf{where\ EU\ in\ LM\ or\ EU\ in\ IN\ or\ EU\ in\ EN} : \mathbf{deny}(\mathbf{EU},\mathbf{O},\mathbf{A})$
$\mathbf{end}$</td></tr>
</table>

Table 6.6 CreateObject access control matrix and location-aware access control policy definition.

<table>
<tr><th></th><th>Local Machine (<b>LM</b>)</th><th>Internal Network (<b>IN</b>)</th><th>External Network (<b>EN</b>)</th></tr>
<tr><td>Resource Owner (<b>RO</b>)</td><td>Allow</td><td>Allow</td><td>Allow</td></tr>
<tr><td colspan="4"><b>policy</b> $P_{LM,IN,EN}(p_{CreateObject})$ ::
$\mathbf{decide}(\mathbf{RO},\mathbf{O},\mathbf{A})$ <b>where RO in LM or RO in IN or RO in EN</b> : $\mathbf{allow}(\mathbf{RO},\mathbf{O},\mathbf{A})$
<b>end</b></td></tr>
<tr><td>Internal User (<b>IU</b>)</td><td>Deny</td><td>Allow</td><td>Deny</td></tr>
<tr><td colspan="4"><b>policy</b> $P_{LM,IN,EN}(p_{CreateObject})$ ::
$\mathbf{decide}(\mathbf{IU},\mathbf{O},\mathbf{A})$ <b>where IU in IN</b> : $\mathbf{allow}(\mathbf{IU},\mathbf{O},\mathbf{A})$
$\mathbf{decide}(\mathbf{IU},\mathbf{O},\mathbf{A})$ <b>where IU in LM or IU in EN</b> : $\mathbf{deny}(\mathbf{IU},\mathbf{O},\mathbf{A})$
<b>end</b></td></tr>
<tr><td>External User (<b>EU</b>)</td><td>Deny</td><td>Deny</td><td>Deny</td></tr>
<tr><td colspan="4"><b>policy</b> $P_{LM,IN,EN}(p_{CreateObject})$ ::
$\mathbf{decide}(\mathbf{EU},\mathbf{O},\mathbf{A})$ <b>where EU in LM or EU in IN or EU in EN</b> : $\mathbf{deny}(\mathbf{EU},\mathbf{O},\mathbf{A})$
<b>end</b></td></tr>
</table>

- *Principle 3*: an internal user can create, read and write to a resource from within the enterprise network, and only read from an external public network.
- *Principle 4*: an external user is always restricted from creating, reading, and writing operations on a resource.

As it follows from these requirements, it becomes important to employ an approach to effectively and efficiently differentiate between various network locations, and – at the same time – combine it with the existing access control policies. Such an approach is expected:

- To be able to detect the current physical/logical location, thus implementing the location-awareness.
- To be able to retrieve and apply corresponding access control policies according to the current physical/network location.
- If the location changes, then apply different policy, thus implementing policy transition.

Such an approach, as described in the previous Chapter, is based on the logical notion of *location* that lies at the core of topological logic.

A simplified illustration of this proposed approach is depicted by the flow chart diagram in Figure 6.1. First, an external access request is generated and intercepted by the proposed system. Next, the system extracts subject and object attributes, including their network locations. It then checks with its internal storage if there are location-aware policies – i.e., policies that hold for specific locations – in its internal repository. If no such policies are found – i.e., an external network location has not been previously registered with the system – an additional user identity check may be invoked (e.g., a two-factor authentication using emails or SMS).[1] Same applies to cases when a new employer's laptop has not been registered in the network yet and is therefore not recognised as a trusted location. This

[1] A potential use of the two-factor authentication goes beyond the scope of the proposed research effort, and we rely on the existing technologies to implement these features if required.

additional check is done to ensure that a legitimate user, albeit attempting to access the access from an unknown remote location/device, will not be rejected and will be able to perform his/her regular working duties. If matching location-aware policies exist, the system checks if the current location of the subject satisfies any of them. If yes, the system then retrieves the referenced SANTA access control policies and evaluates them using the existing SANTA enforcement mechanism, which results in either access being granted or refused. If the location does not match any of the location-aware policies, the access request is rejected for security reasons.

6.4 Case Study Description

This section presents a simple running use case scenario based on the previously described corporate file sharing system and the classification of network locations, users, and their access rights summarised in Tables 6.4,6.5, and 6.6.

Accordingly, the suggested scenario assumes that a user – an employer of the company owning the file sharing system – first creates a file on its local laptop while working on the private corporate network, thereby becoming the ResourceOwner. After editing the file, the user uploads it to an internal corporate server located on the same private network for sharing with the rest of users. After some time, the same user attempts to access the shared file for some further modifications. He first does so while still at work in the office (i.e., from within the private network using a trusted device), and then from home (i.e., from outside the private network, yet a trusted network location, using a trusted device). Also, we assume that at some point the employer travels on a foreign business trip; he first attempts to access the same shared file using his own laptop (i.e., from an untrusted location outside the corporate private network using a trusted device) and then using a public PC located in the hotel the user is currently staying in (i.e., from an untrusted location outside the corporate private network using an untrusted/not registered device).

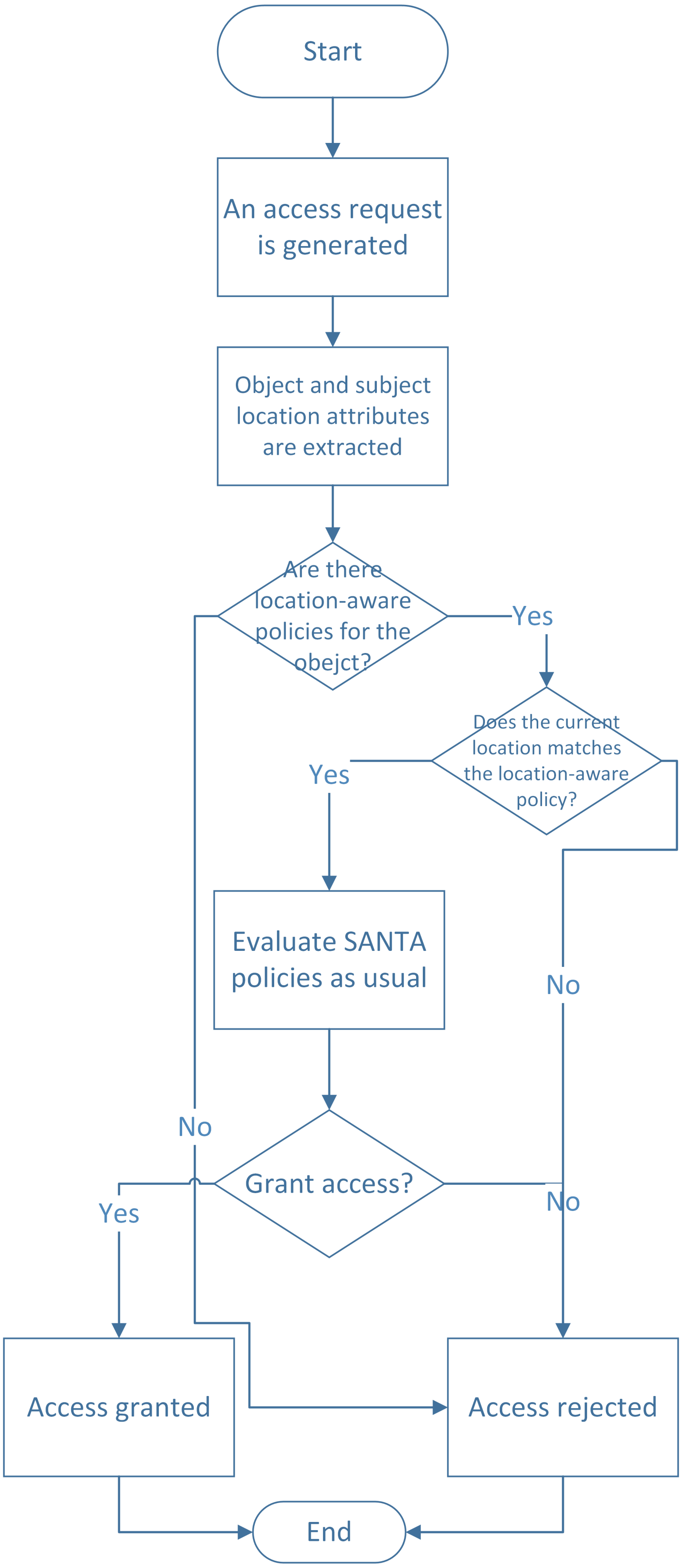

Start
An access request is generated
Object and subject location attributes are extracted
Are there location-aware policies for the obejct?
Yes
Does the current location matches the location-aware policy?
Yes
No
Evaluate SANTA policies as usual
No
Grant access?
Yes
No
Access granted
Access rejected
End

Fig. 6.1 Proposed algorithm combining location-aware policies with SANTA access control policies.

Similar to the ResourceOwner, another employer of the company – i.e., the InternalUser – also attempts to access the shared file first from the private corporate network, then from home, and finally from abroad while on a business trip. Potentially, there could have also been a third type of user involved – i.e., the ExternalUser – but we omit this trivial case, since an external user is always restricted to access internal corporate files.

To handle this manifold scenario, the proposed system has to correctly identify different network locations and apply different access control policies accordingly. The internal operation of the proposed system in each case is summarised in Table 6.7

	Use Case	System Behaviour	Expected System Output for Different Types of Users
1.	Accessing the file located on a private corporate server from the same private network using a trusted device	The system is able to see that the source IP address belongs to the internal network, as well as the MAC address of the user's laptop has been previously registered and is considered as trusted.	Since the file is shared on the corporate network, it is expected to be shared and accessed by all the other users having access to the corporate network. To avoid redundant checks, the access control system immediately grants read and write access for both the ResourceOwner and the InternalUser, skipping the check of user credentials (e.g., an authentication token) – this step is deemed unnecessary due to the completely trusted internal network.

2.	Accessing the file located on a private corporate server from a public, yet trusted network location using a trusted device.	The system intercepts an incoming access control request is able to see that it is coming from an external, yet trusted IP address (i.e., the employer’s home) using a registered trusted device.	Due to the fact that the incoming request comes from outside the private corporate network (albeit the IP address is trusted). Having checked with the PIP, the system now knows that the IP address is trusted, and a corresponding relatively ‘loose’ access control policy is applied. In this case, the system checks the authentication token to validate the credentials of the user. Upon this check, the ResourceOwner is granted read and write access, whereas the InternalUser only gets the read access.
3.	Accessing the file located on a private corporate server from a public and untrusted network location using a trusted device	The system intercepts the incoming request and is able to see that it is coming from an IP address that does not belong to the corporate network. Furthermore, it can see that the IP address actually belongs to a foreign country, and, therefore, more stringent access control policies have to be put in place.	Due to the fact that the request originated from a foreign country, albeit using a trusted device, the system needs to first validate the credentials of the user, and then to enforce a two-factor authentication procedure (e.g., using an email or a text message). Upon the completion of the second authentication step, the ResourceOwner is granted read and write access to the requested file, whereas the InternalUser is only allowed to read the file. In case the two-factor authentication fails, the request is rejected, and the incident might be recorded.

4.	Accessing the file located on a private corporate server from a public and untrusted network location using an untrusted/not registered device.	The system intercepts the incoming request and is able to see that it is coming from an IP address originates from a foreign country. Furthermore, the client device does not belong to the list of previously seen and registered devices maintained by the PIP. Taken together, these considerations make the system apply and enforce the most stringent policies and/or even deny the request.	Due to the fact that the incoming request originates both from an untrusted IP address and unknown client device, the system needs to thoroughly validate the user. More specifically, the system requires the user to go through a three-factor authentication that involves both e-mail and SMS authentication. In case this step is completed successfully by the ResourceOwner, he is granted access to the owned resource. Otherwise, the request is denied, and the incident is reported. The InternalUser is not allowed to do so and is essentially deemed as the ExternalUser in these circumstances; his incoming request may be rejected immediately, even without the additional authentication steps, due to the untrusted nature of both the network location and the device.

Table 6.7 Location transitions handled by the proposed system.

As it follows from these four different use cases, the proposed access control is able to differentiate between different network locations and apply corresponding policies for each situation. It is worth noting that as the level of insecurity increases (i.e., the user leaves the corporate network and access the resource first from home, and then from abroad), the corresponding access control policies get more and more stringent. More specifically, in the described scenario, in the most trusted use case there is no user authentication at all, whereas in the most untrusted use case it includes a three-factor authentication. This way, the system is able to minimise the amount of unnecessary checks when/if possible within the trusted corporate network, and – on the contrary – maximise the security for untrusted network locations. Please note the user authentication in this use case scenario is an example intended to demonstrate the viability of the proposed system, and in practice different access control policies can be put in place.

6.5 System Design and Implementation

A proof of concept implementation has been designed and developed to validate the proposed approach and it is implementation as a software tool. The prototype is based upon the reference architecture, described in the previous chapter, and implements the 5 key conceptual components – namely, Policy Administration Point, Policy Enforcement Point, Policy Information Point, Policy Decision Point, and Policy Retrieval Point. The system prototype has been implemented in Java using the established Eclipse IDE.[2]

6.5.1 System Design

Policy Administration Point

PAP is a component through which policies can be managed – i.e., created, tested, debugged, and modified. For these purposes, it is supposed to have a user interface (e.g., a graphical one or a command line) enable users to interact with the system when managing policies. At the current stage, the system is only implemented as a proof of concept prototype, and does not have a dedicated user interface. Instead, the aforementioned policy management activities can be performed using the default Eclipse IDE functionalities, such as debugging, code assistance, graphical user interface, etc.

[2] https://www.eclipse.org/ide/

Policy Enforcement Point

PEP is the central component of the whole system, as it is responsible for intercepting incoming access requests, passing them further to the PDP for evaluation, and – if access is granted – let the request go through. The main responsibilities, briefly summarised in Chapter 5, are implemented by the PEP as follows. The PEP intercepts incoming network access requests and starts inspecting the packet headers. More specifically, it primarily looks at the IP headers of the source (i.e., the subject attempting to access a resource on the network) and destination (i.e., the object being requested by the subject). For the private corporate network, the PEP also extracts the source MAC address of the original device (which is not possible for a request coming from the public Internet, since the original MAC addresses get overwritten while being transferred from one broadcast domain to another). Having obtained these subject and object attributes, the PEP forwards this information to the PIP that will check if there are corresponding policies associated with the provided addresses, and together with the PDP will evaluate these policies with respect to the extracted addresses. Once the evaluation is complete, the PEP will receive the policy evaluation decision and apply (i.e., enforce) this decision via the PEP. That is, it will either let the incoming request go through – in case the decision is positive, or will reject it – in case the decision is negative. As it was also mentioned, a two-factor authentication may be potentially applied here as well in order to avoid situations when a legitimate user cannot be granted access due to a not yet registered laptop or a network location. Otherwise, the PEP will reject the request and notify the subject with a corresponding error message. From this perspective, it can be seen as the coordinator of the whole access control procedure, as it manages the triggers the enforcement process and manages interaction with other components of the system.

Policy Information Point

PIP is responsible for implementing the functionality related to location-awareness. To achieve this, the PIP is able to receive the incoming access request from the PEP and extract location-related information. This is implemented using the standard WHOIS client library from Apache and the third-party GeoIP Legacy Java API.[3] Essentially the GeoIP library provides a collection of methods for querying a constantly updated database, in which network IP addresses are mapped to their geolocations. A code snippet of the PIP functionality, retrieving locations is included in Listing 6.1. The function `getLocation` takes as input an IP address and a file with records containing the geolocation information (provided by the GeoIP API developers). As an output, the function returns an object of the class `ResourceLocation` that contains all the geolocation-related information (i.e., country code, country name, region, city, postal code, as well as altitude and longitude), required for taking a policy enforcement decision.

Listing 6.1 Obtaining location from an IP address.

```
public ResourceLocation getLocation(String ipAddress, File dbase) {
  ResourceLocation resourceLocation = null;
  try {
    ResourceLocation = new ResourceLocation ();
    LookupService lookup = new LookupService(dbase,
                                LookupService.GEOIP_MEMORY_CACHE);
    Location locationServices = lookup.getLocation(ipAddress);
    resourceLocation.setCountryCode(locationServices.countryCode);
    resourceLocation.setCountryName(locationServices.countryName);
    resourceLocation.setRegion(locationServices.region);
    resourceLocation.setRegionName(regionName.regionNameByCode(
                       locationServices.countryCode,
                       locationServices.region));
```

[3] https://github.com/maxmind/geoip-api-java/

```
    resourceLocation.setCity(locationServices.city);
    resourceLocation.setPostalCode(locationServices.postalCode);
    resourceLocation.setLatitude(String.valueOf(
                       locationServices.latitude));
    resourceLocation.setLongitude(String.valueOf(
                       locationServices.longitude));
    } catch (IOException e) {
      System.err.println(e.getMessage());
    }
  return resourceLocation;
}
```

Having identified the location of the object/subject, the PIP is now able to evaluate the access control policies concerning the requested object. More specifically, it first checks for location-aware policies that hold, given the current locations of the subject and the object. If there are such policies present, the PIP proceeds with the actual evaluation of the SANTA policies that are referenced by the location-aware policies.

Policy Decision Point

PDP is responsible for the actual decision taking based on combining SANTA access control policies – on one hand, and network locations – on the other. This component actively interacts with the PRP and PIP, since whenever there is an incoming access control request, it requires fetching *i)* SANTA policies associated with the requested object (from the PRP), *ii)* location-aware policies associated with the retrieved SANTA policies (from the PRP), and *iii)* network locations of the object and the subject, involved in the current scenario (from the PIP). As a result, based on the object/subject current network locations, the PDP is able to decide whether the access has to be granted or not.

Policy Retrieval Point

PRP is the central repository for storing and retrieving policies, as well as previously recorded network locations. In the current prototype implementation, a simple MySQL database has been used to enable the required functionality. In the future, however, a more advanced and sophisticated solution (e.g., a NoSQL database) might be considered.

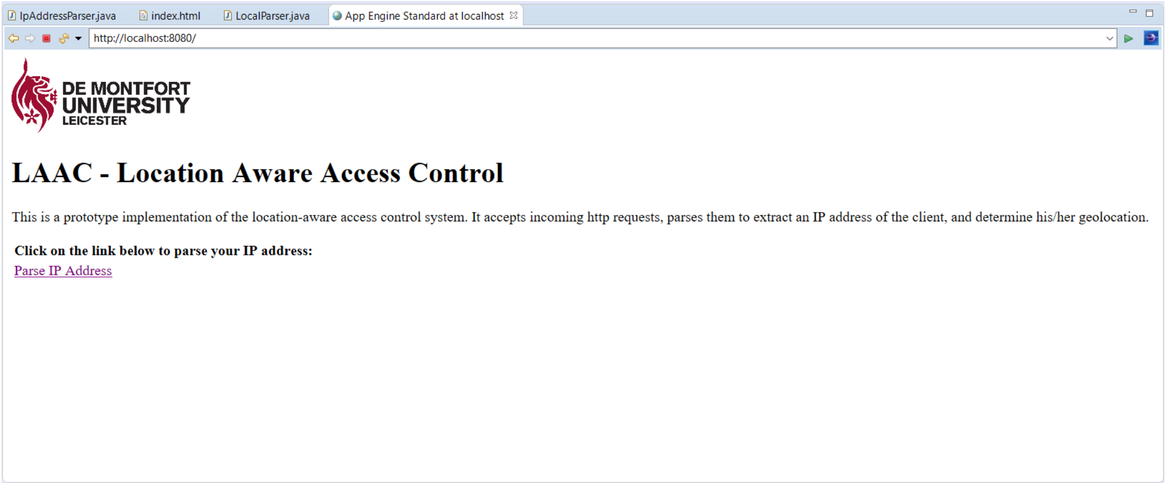
6.5.2 System Implementation and Operation

The above described design has been implemented as a Web application (i.e. Java servlet application), deployed and running on the Google App Engine cloud.[4] It has a REST API which can be accessed remotely by users from different geographical locations. The system then parses incoming requests, extracts IP addresses, and determines precise geophysical location of clients. This way, it is possible to grant or deny access to resources based on the current location of users. The application uses GeoIP[5] – an existing library for discovering information about a specific IP address. The library provides a database of IP addresses mapped to precise geophysical locations. Based on the identified location, the system then decides whether access should be granted. Figure 6.2 demonstrates the index page of the developed system, whereas Figures 6.3, 6.4, 6.5, and 6.6 illustrate how the system is able to extract geophysical location of incoming requests, and decide whether access should be granted. For In the screenshots below, client requests from China and Russia are considered untrusted, whereas requests from the UK (Leicester and London) are trusted.

It is also important to detect situations when a user transits from one location to another, which possibly has different access control policies. In such situations, to identify user devices across different incoming requests (and thus be able to apply different access control policies), the system keeps track of devices' MAC addresses. This way, it is possible to

[4] https://github.com/nasserabwnawar/laac

[5] https://dev.maxmind.com/geoip/



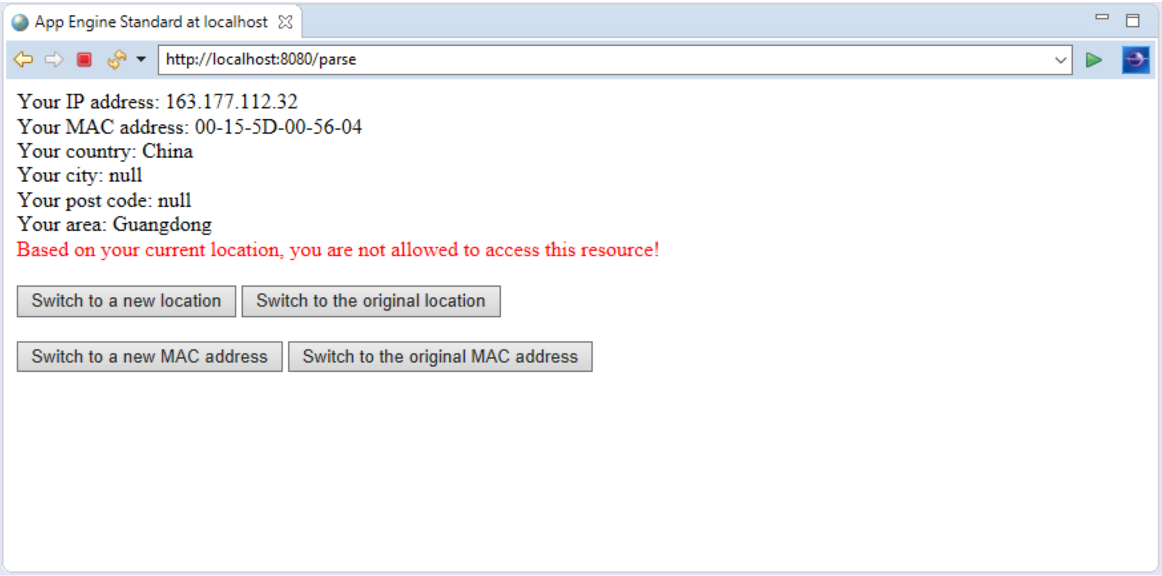
LAAC - Location Aware Access Control

This is a prototype implementation of the location-aware access control system. It accepts incoming http requests, parses them to extract an IP address of the client, and determine his/her geolocation.

Click on the link below to parse your IP address:

Parse IP Address

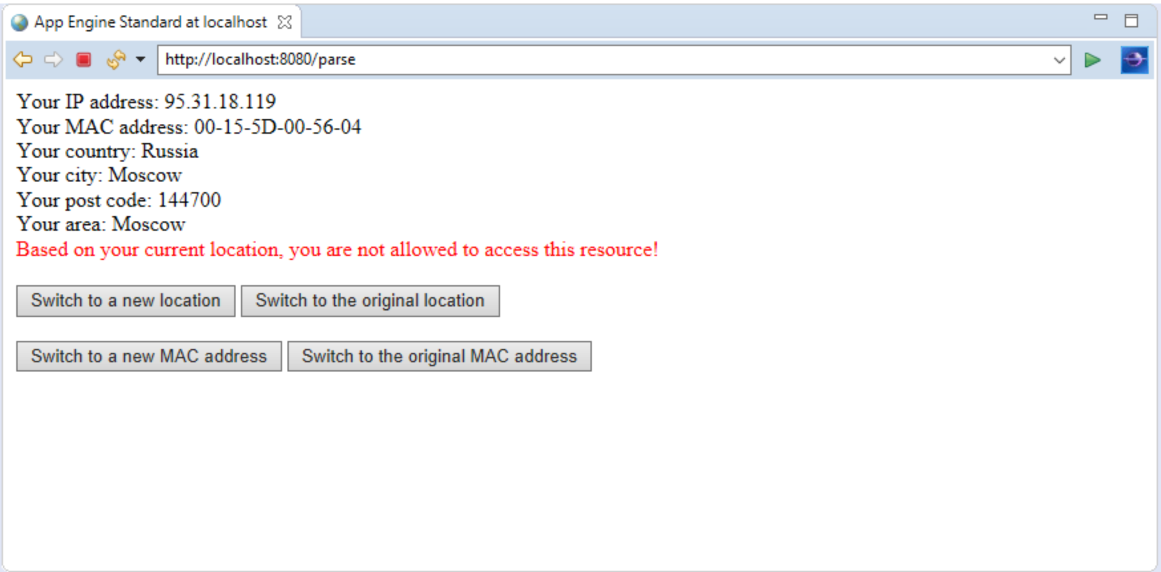
Fig. 6.2 Home page.


App Engine Standard at localhost
http://localhost:8080/parse

Your IP address: 163.177.112.32
Your MAC address: 00-15-5D-00-56-04
Your country: China
Your city: null
Your post code: null
Your area: Guangdong
Based on your current location, you are not allowed to access this resource!

Switch to a new location
Switch to the original location
Switch to a new MAC address
Switch to the original MAC address

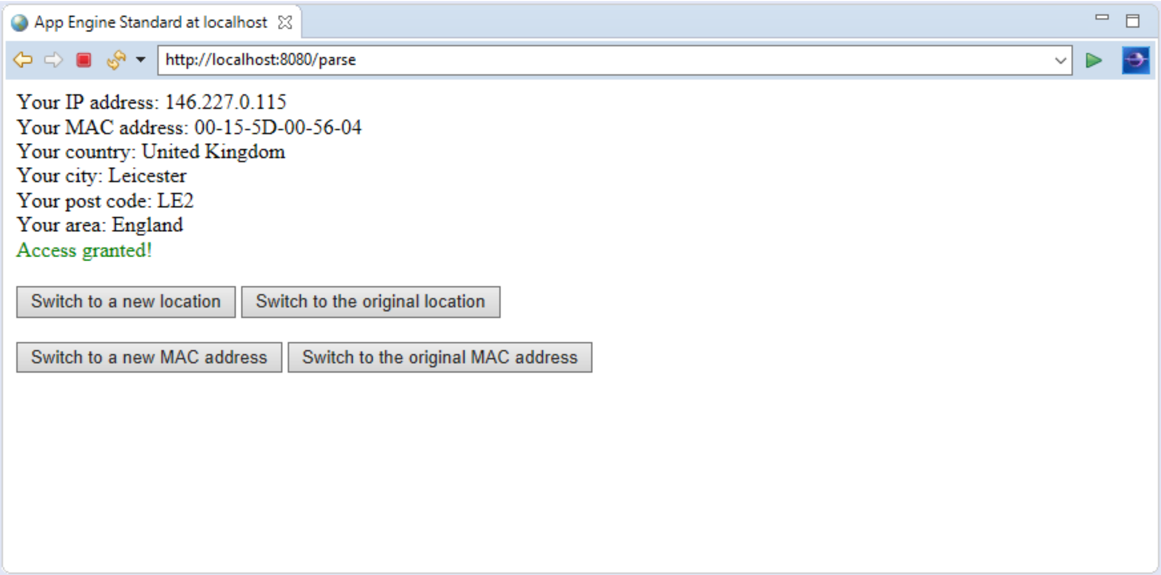
Fig. 6.3 Sample request from an untrusted location (China).


App Engine Standard at localhost
http://localhost:8080/parse

Your IP address: 95.31.18.119
Your MAC address: 00-15-5D-00-56-04
Your country: Russia
Your city: Moscow
Your post code: 144700
Your area: Moscow
Based on your current location, you are not allowed to access this resource!

Switch to a new location
Switch to the original location
Switch to a new MAC address
Switch to the original MAC address

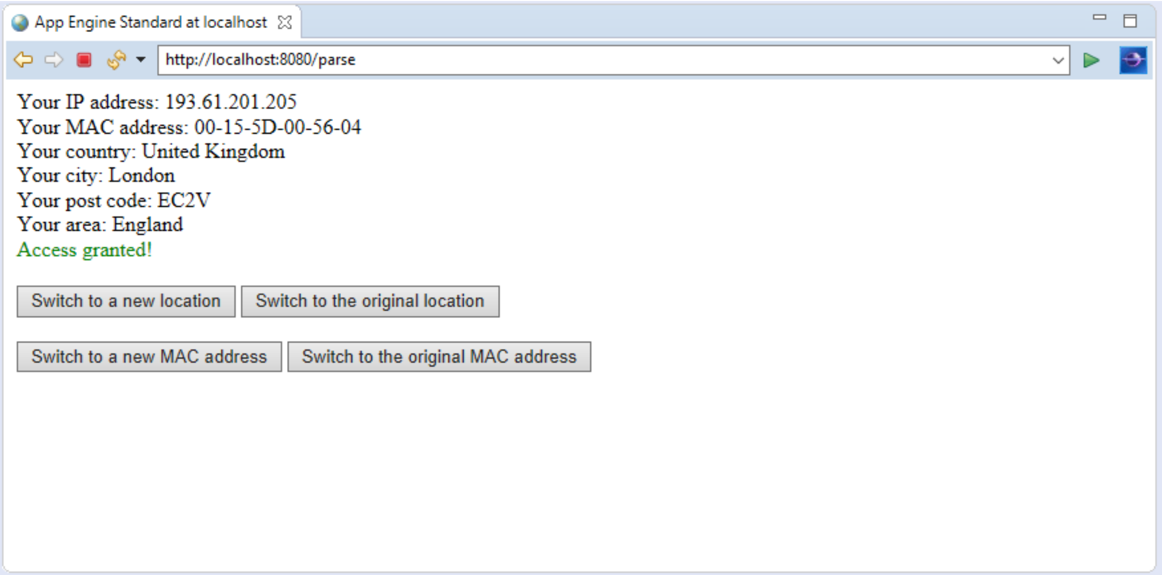
Fig. 6.4 Sample request from an untrusted location (Russia).


App Engine Standard at localhost
http://localhost:8080/parse

Your IP address: 146.227.0.115
Your MAC address: 00-15-5D-00-56-04
Your country: United Kingdom
Your city: Leicester
Your post code: LE2
Your area: England
Access granted!

Switch to a new location
Switch to the original location
Switch to a new MAC address
Switch to the original MAC address

Fig. 6.5 Sample request from a trusted location (Leicester, UK).


App Engine Standard at localhost
http://localhost:8080/parse
Switch to a new location
Switch to the original location
Switch to a new MAC address
Switch to the original MAC address

Your IP address: 193.61.201.205
Your MAC address: 00-15-5D-00-56-04
Your country: United Kingdom
Your city: London
Your post code: EC2V
Your area: England
Access granted!

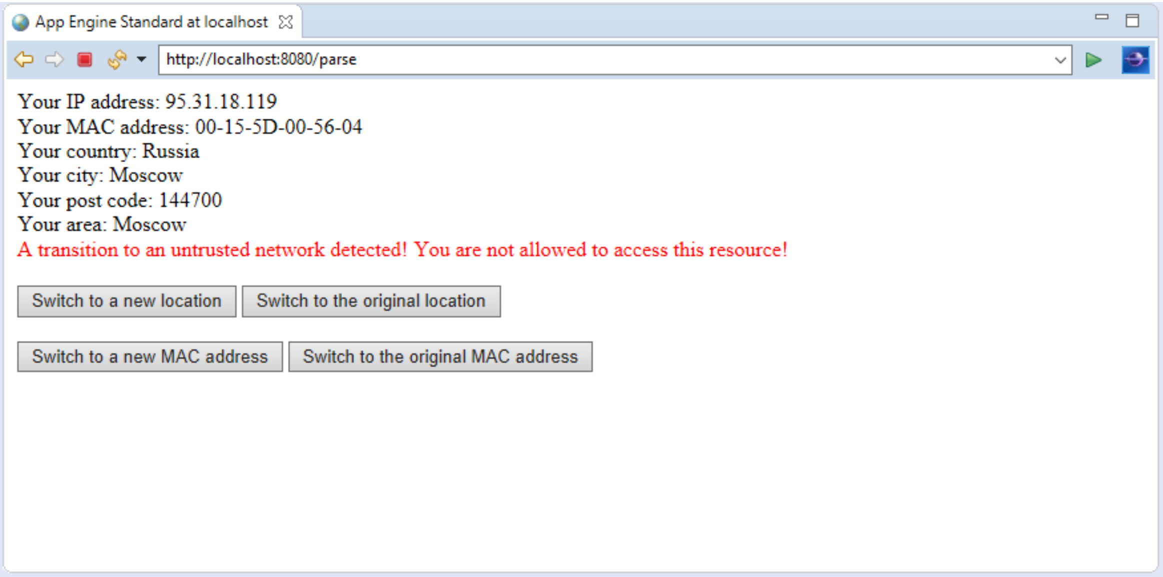
Fig. 6.6 Sample request from a trusted location (London, UK).

check situations when a device transitions from one network to another, and therefore access control policies should be re-evaluated. Similarly, if there is an incoming request with a different MAC address, it is treated as a new device and requires access policy re-evaluation.

In Figures 6.7 and 6.8, for simplicity, network transitions and MAC address changes are triggered by clicking the corresponding button. For experimental purposes, it is also possible to revert to the original network location and MAC address. The screenshots demonstrate how after a network transition, access policies are re-evaluated to continue permitting or denying access to a resource.

Experiment setup and benchmarking

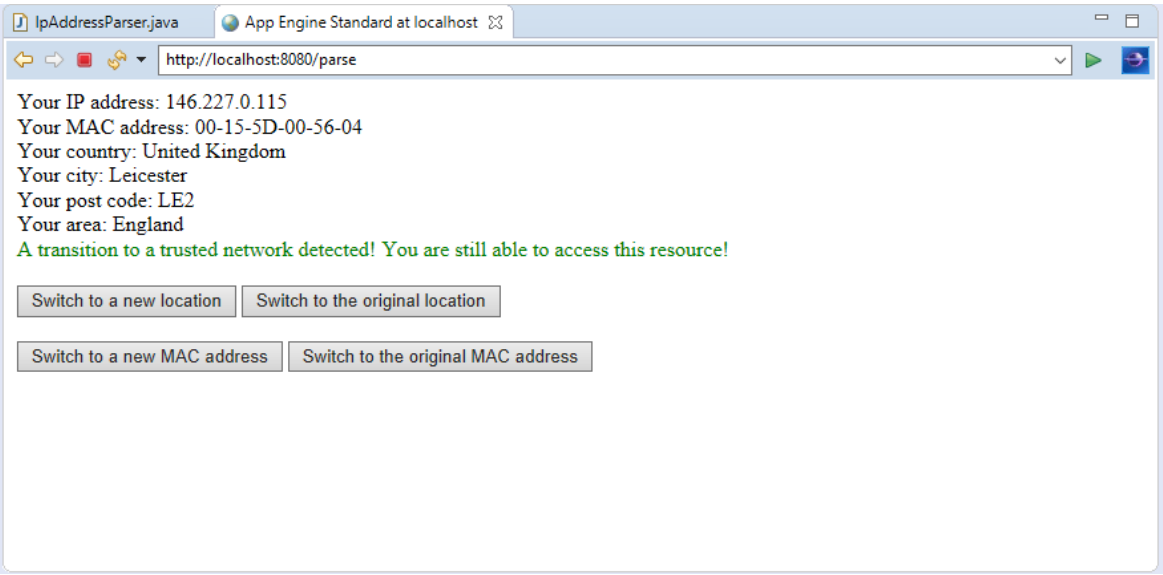
The main goal of the presented experiments is to demonstrate the performance "footprint" of the developed prototype. That is, the experiments show how much overhead is caused by preforming this additional location-aware check when evaluating access control policies. As previously discussed, unnecessarily strict access control policies required relatively expensive computation may be reduced when an access request is originating from a trusted location.


App Engine Standard at localhost
http://localhost:8080/parse

Your IP address: 95.31.18.119
Your MAC address: 00-15-5D-00-56-04
Your country: Russia
Your city: Moscow
Your post code: 144700
Your area: Moscow
A transition to an untrusted network detected! You are not allowed to access this resource!

Switch to a new location
Switch to the original location
Switch to a new MAC address
Switch to the original MAC address

Fig. 6.7 A transition to an untrusted network is detected.


IpAddressParser.java
App Engine Standard at localhost
http://localhost:8080/parse

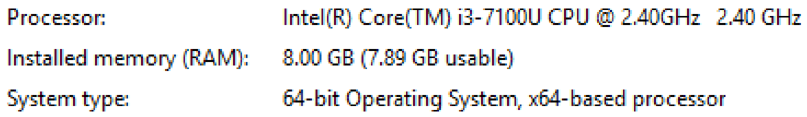
Your IP address: 146.227.0.115
Your MAC address: 00-15-5D-00-56-04
Your country: United Kingdom
Your city: Leicester
Your post code: LE2
Your area: England
A transition to a trusted network detected! You are still able to access this resource!

Switch to a new location
Switch to the original location
Switch to a new MAC address
Switch to the original MAC address

Fig. 6.8 A transition to a trusted network is detected..

From this perspective, performing the location-aware check, albeit introducing an additional delay, may improve the overall performance of a location-aware access control system.

To evaluate the performance of the implemented network location detection solution, we used and existing IP address dataset provided by the GeoIP library. The dataset contains over 150,000 records, which were used to test the performance of the proposed implementation. The hardware setup of a machine on which the experiments were executed are summarised in Figure 6.9.



Processor:	Intel(R) Core(TM) i3-7100U CPU @ 2.40GHz 2.40 GHz
Installed memory (RAM):	8.00 GB (7.89 GB usable)
System type:	64-bit Operating System, x64-based processor

Fig. 6.9 Testbed hardware setup.

Therefore, the main benchmarking metric was time delay – a time difference between the moment when a user request is received by the server and the moment when an access control decision, based on policies, it is eventually generated. The obtained results are depicted as a chart in Figure 6.10. The chart contains two bars, each representing a separate set of experiments. The light blue bar depicts experiments, where no proposed location-aware functionality was used (i.e. upon receiving a user request, the system start evaluating access control policies in place), whereas the dark blue bar corresponds to the use of the proposed functionality. In this case, the delay also includes time required to parse the request, extract location-specific information, convert into a precise geophysical location, and select a corresponding policy to be applied. Only after that the policy is evaluated and access control decision is generated.

As it follows from the chart, the performance overhead of the implemented system using the current data set is less than a second (i.e. 797 milliseconds). In relative terms, this 66% increase seems to be quite a considerable difference. However, in practice, given the critical importance of the access control domain, this additional overhead comes at a cost of more precise access control decisions.

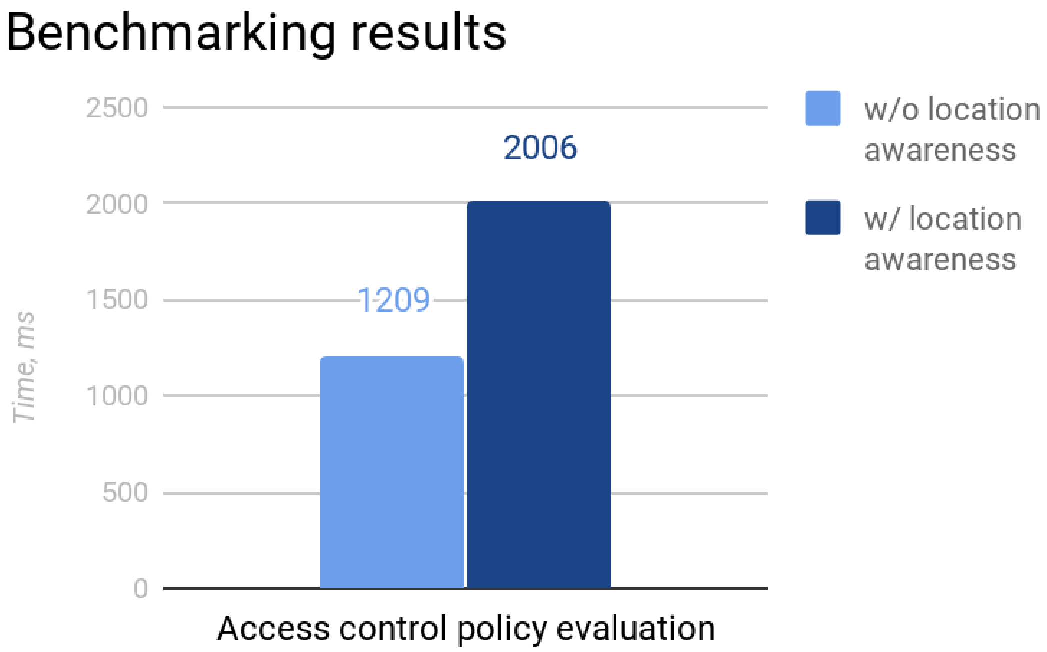


Fig. 6.10 Benchmarking results.

Besides, as was previously discussed, the overall time delay may further go down, because less computationally-intensive access control policies are evaluated for trusted locations. For example, by applying a simple, yet efficient approach and blacklisting certain locations (i.e. countries), which are traditionally considered to be untrusted, it is possible to considerably reduce the system time required to perform access control evaluation. More specifically, in the experimental data set consisting of **150,000** records with IP addresses mapped to geophysical locations, **6,845** records referred to Russia and **3,721** records referred to China. As a result, by blacklisting the two countries, the system was able to reduce the number of access control invocations by 7%. Applying a more fine-grained and differentiated approach and evaluating the system on a set of policies, which vary in their complexity, depending on the user location, is planned as another direction for future work.

Finally, the presented benchmarks only represent the current implementation, which primarily remains a research proof-of-concept prototype, which is yet to be optimised. In future, it is expected that the system will follow best software engineering practices, and deployed on more powerful hardware to demonstrate the full potential of the proposed approach.

6.6 Summary

Based upon the materials presented in the previous chapter, this chapter described the actual implementation details and the underpinning logical foundations of the proposed approach. More specifically, to implement the described location-aware functionality, the presented work adopted the principles of Topological Logic presented in Chapter 4, which enables to differentiate between various different logical and physical locations, and, thereby, check if specific propositions hold for specific locations. Speaking in simpler words, by applying the Topological Logic principles, it is possible to use the information about the original location of an access request in order to apply corresponding SANTA access control policies.

As a result, it is possible to achieve a flexible and fine-grained control over access control with respect to different network location, without duplicating policies. This has also been demonstrated through an implemented prototype system and a running case study, focusing on a corporate file sharing system. A more detailed analysis and discussion of the main benefits of the presented approach, as well as its potential shortcomings will be provided in the next chapter, which will also re-visit the introduction of this thesis to see whether the presented research has addressed all of its initial research goals and objectives.

Chapter 7

Analysis and Discussion

Objectives:

- To evaluate whether outlined research question and research goals have been addressed.
- To summarise main benefits of the proposed approach, as well as its potential shortcomings.
- To summarise main contributions of the thesis.

7.1 Introduction

The final chapter of this thesis summarises all the materials, which have been described so far. It lists and explains potential benefits associated with the approach, as well as its potential limitations, in a more structured manner. So far, the potential benefits and shortcomings of the proposed approach have been spread across several chapters and sections. In this chapter, our goal is to explicitly bring together and evaluate the main potential benefits and limitations of our proposed approach. In summary, the potential benefits include:

- Declarative approach to defining policies and the separation of concerns.
- Novel way of capturing the spatial dimension in access control, thus combining the temporal and spatial dimensions.
- Increased level of reliability and automation underpinned by the underlying logical formalisms (i.e., Interval Temporal Logic and Topological Logic).
- More optimised utilisation of computational resources due to minimisation of unnecessary access control checks.
- Potential to complement existing approaches.

The description of our approach would not be completing without discussing potential limitations, which include:

- Lack of real-world implementation/deployment and experiments.
- Lack of deeper integration with SANTA – i.e., the proposed location-aware concepts are not part of the SANTA language suite yet, but rather complement it as an independent component.

The chapter also concludes the whole thesis with an overall summary of the presented ideas. It evaluates the accomplishment of the goals and tasks, outlined in the introductory chapter of the thesis, and once again summarises its main contributions.

7.2 Evaluating the results: main benefits

We now summarise what has been already explained so far, in a more structured manner. First, we list potential benefits of the proposed approach so as to explain why each particular feature of the approach is of benefit – that is, how and what problem it solves.

7.2.1 Declarative approach to defining policies and the separation of concerns

One of the main motivating factors and goals of the research work described in this thesis was the creation of a mechanism, which would:

- Separate the definition of location-aware policies from the actual enforcement of these policies.
- Allow the definition of the policies in a declarative, loosely-coupled manner.

The first requirement on its own can be simply addressed with traditional, component-based programming techniques. That is, one can capture knowledge at the level of the programming source code in a separate component (e.g., a class, library, etc.). The second requirement, however, calls for applying more sophisticated techniques for policy definition. Using the Topological Logic as an underlying logical formalism can successfully address this requirement, by providing us with the possibility to declaratively define location-related aspects in access control policies. This way, it will become possible to modify them at dynamically run-time if/when required, without recompiling and restarting the whole application system. In other words, with policies separated from the platform/application programming code, it is easier to make changes 'on the fly' and to maintain the whole system in a stable, operational state.

7.2.2 Novel way of capturing the spatial dimension in access control, thus combining the temporal and spatial dimensions

One of the main contributions of the presented research work is the concept of location-awareness in the context of access control policy enforcement, which complements the existing temporal dimension of access control implemented by the SANTA language and the

Interval Temporal Logic. So far, there has been little evidence in the literature of combining the temporal and spatial dimensions at the level of logical formalisms in the context of access control policies. In these circumstances, the proposed approach may potentially open new opportunities for creating a next generation access control system, where the two dimensions are equally important. For example, it may become possible to create multi-factor authentication to enable access to requested resources, using not only users' credentials, but also their current location and time – a mechanism, which is currently beyond the default functionality of the traditional access control systems.

7.2.3 Increased level of reliability and automation underpinned by the underlying logical formalisms

By using the existing reliable logical formalisms, such as Interval the Temporal Logic and the Topological Logic, the proposed approach is expected to benefit from increased reliability and automation. That is, instead of 're-inventing' the wheel and developing the underlying access control policy enforcement with numerous 'if-then' operators in a hard-wired manner, the security/access control engineers can rely on the existing functionality and powerful logical reasoning.

7.2.4 More optimised utilisation of computational resources due to minimisation of unnecessary access control checks

The proposed approach enables differentiating between various physical and logical locations, and thereby treat them differently in terms of access control. That is, it is create some kind of location profile by mapping locations to different level of threat and risk. This way, depending on the location profile, the access control enforcement mechanism can apply a corresponding set of policies. For example, if there is a previously unseen incoming request from a country/region associated with an increased rate of cyber-crime, a right decision will be to apply more stringent access control policies. Similarly, if the location, where an access request is coming from, is well known and trusted, access control policies can be released to some extent or disabled completely (e.g., for the internal corporate network). As a result, such a fine-grained application of policies leads better utilisation of computational resources. By minimising the amount of unnecessary security and access control checks in situations, when they are not really required, it is possible to save a considerable amount of computational resources, which will only be used when truly needed.

7.2.5 Potential to complement existing approaches and languages

As it was described, the presented location-aware approach to enforcing access control policies was combined with the existing policy language SANTA, which is not the only language to define access control policies. Implemented in a loosely-coupled manner, the proposed system can be potentially combined with other existing access control approaches and languages in a similar way. That is, in situations, when it is required to gather location-specific information and then react with respect to this information, it is possible to integrate the presented location-aware functionality. From this perspective, the presented approach is seen as complementary to the existing works.

7.3 Evaluating the results: potential shortcomings

Next, we continue with potential limitations of the approach, which are listed and summarised in a similar manner. When talking about limitations, we are mainly discussing shortcomings, i.e., features which are currently beyond our capabilities to be addressed. In contrast, improvements, which can be done but due to certain reasons (e.g., time constraints or being not directly relevant to the described PhD research topic) have not been addressed yet, are included in the concluding chapter as directions for future work.

7.3.1 Lack of large scale real-world implementation/deployment and experiments

In the previous section, we described a case study, aiming to demonstrate how the proposed approach can be potentially applied and used. The case study, however, albeit based on an existing cloud-based file sharing application, is primarily hypothetical. In these circumstances, the evaluation of the results is somewhat constrained with the assumptions of the described use case scenario. There may well exist circumstances that the modelled use case environment is unable to reflect in existing systems. However, as a proof-of-concept the prototype has demonstrated that the approach presented in this thesis is viable and scaling up to a full system should not present an issue. To further address this limitation, a more in-depth and realistic experimentation is included as part of the future work.

7.3.2 Lack of native integration with SANTA

As it was explained, at the moment the presented location-aware functionality relies on manual extraction of subject physical and logical locations from IP addresses to be further used during the policy evaluation and enforcement procedure. That is, the actual access control policies are defined using the plain SANTA language, whereas the system takes location into consideration only at the level of programming code. That is, at the moment, SANTA is not yet equipped with location-specific constructs and operators, which would enable access control engineers to capture this logic straight inside access control policies. From this perspective, it can be said that SANTA has not yet been extended so as to offer native support for location-awareness in the context of access control.

7.4 Answering the research question and meeting goals

Following the motivation of insufficient support for applying location-aware access control policies, which would differentiate between different location where access control policy objects and subjects are currently located, in the introduction to this document we have raised and formulated the main research question to be addressed and answered by the presented PhD work:

"How to enable heterogeneous computing systems, spanning across multiple physical and logical locations, as well as different administrative domains and ownerships, with support for location-aware access control policy enforcement, and implement a differentiated fine-grained access control depending on the current location of subjects and objects?"

To address this research question, we have put forward a hypothesis that the outlined challenges can be potentially addressed by extending the existing functionality of access control tools and languages, such as SANTA, with native support for detecting the current location of protected resources, as well as of users trying to access them, and thereby enabling these existing tools to apply and enforce differentiated fine-grained access control policies with respect to the current locations. Accordingly, we have proved the outlined hypothesis and answered the main research question by proposing the concepts of location, location awareness, and location transition, and devising a location-aware access control mechanism, underpinned by these novel concepts.

To provide a more structured overview of the presented research, we have also broken down the main research goal into several theoretical, technical and experimental objectives, which have been successfully fulfilled, and are now discussed in more details below.

7.4.1 Meeting theoretical objectives

- We have studied the state of the art in the domain of cloud computing with a specific focus on existing challenges, especially in the context of access control in heterogeneous hybrid cloud environments (see Chapter 3).

- We have identified an existing research and technological gap to be addressed by the proposed research – namely, a lack of support for location-aware access control policy enforcement, which results in an inflexible and coarse-grained architectures.

- We have proposed a potential approach to address the identified challenges – as described in the corresponding chapters (see Chapters 5 and 6), we have tackled these challenges by introducing the concepts of location and location-awareness that underpinned the access control policy transition and enable location-specific policy enforcement.

7.4.2 Meeting technical objectives

- We have designed and implemented a software prototype of a location-aware access control mechanism that is able to extract information related to users' and resources' locations based on IP addresses, and thereby enable policy transition and differentiated policy enforcement (see Chapter 5). The core access control policies were implemented using SANTA – an existing access control policy language, whereas the novel location-aware functionality was implemented from scratch.

- In Chapter 6, we have designed and implemented a hypothetical case study that highlights existing challenges and gaps in the domain of access control in heterogeneous (cloud) environments, aiming to prove the presented concepts and demonstrate the viability of the whole proposed approach.

7.4.3 Meeting experimental objectives

- We have designed and developed a prototype access control system that combines the existing SANTA functionality and the novel location-aware features (see Chapter 5). The latter are formally underpinned by the topological logic.
- We have designed and conducted a hypothetical case study that served to validate the proposed approach and to test the implemented prototype (as described in Chapter 6). The case study is based on a corporate cloud-based file sharing system that involves multiple network locations for storing and accessing files. As demonstrated by the case study, the approach can detect users' current locations and apply corresponding access control policies respectively.

7.5 Summarising contributions

This section revisits the main contributions of the proposed approach to provide the reader with a better understanding of the author's achievements in the context of the presented PhD research. Achieving the goals of the proposed approach primarily contributes to the research areas of computer security and access control – in general, and access control for heterogeneous/hybrid environments – in particular. The approach puts forward the novel concepts of access control object/subject location as well as location-awareness as a key characteristic of a policy enforcement mechanism responsible for evaluating policies and taking access control decisions. Neither of these features have been previously proposed, discussed or implemented, which makes the proposed approach a potentially valuable contribution to a wide range of academic researchers and industrial practitioners. Moreover, the contribution of the thesis also spans across several adjacent research fields, such as Cloud Computing (and especially – Hybrid Cloud Computing), as well as provides a new application domain for the existing logical formalisms, including Interval the Temporal Logic and Topological Logic.

More specifically, the main contributions of the described research effort can be summarised as follows:

- *Literature survey of the state of the art in access control in (hybrid) cloud computing* – as part of fulfilling the theoretical objectives, a literature survey has been conducted, identifying existing limitations and gaps in the considered research field. As it became clear, the challenging topic of insufficient support for taking into consideration subject and object location when enforcing access control policies is yet to be explored. The conducted survey, as well as the whole presented research work in general, is intended to raise the overall awareness within the research community and attract further attention to this motivating and challenging problem.

- *Definition of functional requirements for a location-aware access control system* – based on a thorough investigation of the existing access control systems, approaches, and techniques, some existing limitations have been identified. These limitations, in turn, led to devising a list of functional properties for an envisaged solution. Briefly, these include support for modelling and extraction location-related information, as well as an ability to enforce access control policies with respect to physical and logical locations of both subjects and objects. This functional specification serves as the key reference underpinning the design and implementation of the future system. Moreover, it also contributes to the state of the art in enabling location-aware access control policy enforcement, as it is expected to be re-used by the wider research community, willing to engineer their own solutions based on the proposed approach (i.e., and thus not 're-inventing the wheel').

- *Novel concepts of location, location-awareness and policy transition* – these novel concepts have been proposed to address the requirement of enabling differentiated treatment of resources and users depending on their contexts. More specifically, it was important to introduce and clearly define the main concepts, so as to be able to further build the whole approach based on them. These concepts are seen as contributions, because previously there has been little evidence of integrating the spatial dimension into the context of access control policy enforcement. Discussing these concepts in this thesis will hopefully contribute to creating next-generation logically-underpinned access control systems in the future.

- *Design and prototype implementation of the location-aware access control system* – using the proposed system, one is expected to benefit from the possibility to enable location-aware access control policy enforcement. Moreover, the outlined functional specifications underpinned the conceptual design of the proposed system. In the future, it has the potential to act as a reference model for the wider research community, who are willing to implement their location-aware access control policy enforcement. As far as the prototype implementation is concerned, we have developed a prototype version of the proposed system, which serves to demonstrate the viability of the whole presented approach. Using this system, users are expected to benefit from the possibility to enable location-aware access control policies in heterogeneous computer environments. Moreover, since the current implementation follows an open-source approach to software development and distribution, users are also encouraged to further extend the existing functionality to implement required emerging features, and thereby act as contributors to our system.

7.6 Summary

This chapter served to summarise the main results of the presented research effort. It first summarised and discussed the main benefits of the presented research and the developed location-aware access control system. Among these benefits are the declarative approach to defining policies and the separation of concerns, the novel way of capturing the spatial dimension in the context of access control, the increased level of reliability and automation, more optimised utilisation of computational resources, and the potential to complement existing approaches. The chapter also summarised the potential shortcomings of the approach, among which the lack of real-world deployment and validation, as well as insufficient integration with the SANTA language, are primarily highlighted. Next, the chapter evaluated the conducted work with respect to the initially outlined thesis goals and objectives, as well as the main research question, and concluded that all the goals have been met, the main research question has been answered, and the hypothesis has been proved.

The list of potential shortcomings of the presented approach only includes the aspects, which cannot be easily addressed at the moment. Those shortcomings or potential directions for improvements, which can be addressed and implemented, are summarised in the next chapter as part of future work. The next chapter concludes the overall thesis with some final remarks and discussions.

Chapter 8

Conclusion and Future Work

Objectives:

- To summarise and conclude the thesis.
- To discuss potential directions for future work.

8.1 Introduction

This very last chapter of the thesis serves to conclude the whole thesis by highlighting the motivation behind the presented research effort and summarising the key research findings and contributions. The chapter briefs the reader on the several potential directions for future work, which can potentially further extend the existing system.

8.2 Thesis overview

The presented research work looked into the pressing challenge of insufficient support for location-aware access control. As distributed software systems exponentially grow in size and complexity, they start spanning across multiple geographical locations (e.g., cities, regions, and even countries), as well as several ownerships and administrative domains. Furthermore, modern software, following the service-oriented principles, is designed and implemented to simultaneously serve multiple users in a multi-tenant manner. The increasingly challenging issue of access control becomes particularly apparent in the case of hybrid cloud environments, which assume that different components of complex software systems are spread across different network locations, including private enterprise networks and public cloud platforms. The dynamic nature of such hybrid scenarios requires that sensitive information, being transferred across different network locations, is protected from unauthorised/malicious access by a corresponding access control mechanism at all times. As it was revealed, existing approach typically do not take into consideration the spatial dimension when evaluating and enforcing policies, resulting in non-optimised and inefficient utilisation of resources. That is, unnecessary access control check is performed in relatively safe contexts, whereas in truly risky location they are insufficient.

To address this limitation, in this thesis we presented a location-aware approach to access control, which enables flexible, fine-grained policy enforcement with respect to the current locations of requested resources and users, requesting them. To achieve, this the approach takes the existing access control policy language SANTA, which is based on the Interval Temporal Logic, and combines it with the Topological Logic, thereby creating a two-dimensional solution covering both the temporal and the spatial dimensions. As demonstrated by a hypothetical case study, based on a distributed cloud-based file sharing and storage system, deployed and used within an enterprise, the proposed approach has the potential to address the outlined research challenges and advance the state of the art in the field of access control in distributed heterogeneous digital environments, such as hybrid clouds.

8.3 Future work

In this section, we explain possible directions for further research and explain what else can be explored and implemented in order to make the presented approach even more efficient and effective. Some of these directions go beyond the scope of this presented research effort, and are worth a PhD research in their own right.

- *Evaluation and experimentation on a real-life hybrid cloud use case*: the described case study, albeit demonstrated the general viability of the proposed approach, is based on multiple assumptions, which limit the wider applicability of the developed framework. It is, therefore, desirable to further develop the prototype and to deploy it in real-life ICT settings within an enterprise. As a potential extension to the existing case study, we might consider an assignment submission system of a university. Such a system is characterised by multiple network locations (potentially including remote cloud-based ones), where 'sensitive' information (i.e., student assignments, marks, personal information, etc.) is stored, and which differ in terms of their access control 'sensitivity'. Similarly, multiple users can access the submitted resources either from their personal devices or university-owned stations from within the internal university network, or from outside using personal devices. Taken together, these factors might constitute a valid case study to further evaluate the presented approach.

- *Extending the SANTA vocabulary with location-aware operators*: as highlighted in the list of potential shortcomings, at the moment the presented location-related information is extracted from the user requests and resources network addresses, and then used by the policy enforcement component when evaluating policies. In the future, we are planning to extend the existing SANTA vocabulary to equip it with location-aware operators. That is, it will be possible to specify location-specific constraints within access control policies themselves, rather than specifying this location-aware logic in the source code of the policy enforcement mechanism. This will be possible thanks to the combination of the two types of logics – namely, the Interval Temporal Logic, currently underpinning the SANTA language, and Topological Logic that allows formally define network locations. As a result, a declaratively-defined and loosely-coupled architecture will be achieved, such that the actual access control logic will be solely defined using the policy language, whereas the enforcement of the policies will be implemented in the source code. This way, it will be possible to modify policies in a transparent and non-intrusive manner, without massive source code modifications and system reboot.

- *Implementing other ways of extracting location-related information*: in the current version of the prototype implementation, for demonstration purposes, we employed a rather simplistic and naïve approach to extract and collect information about user and resource location, based on extracting network IP addresses and mapping them to actual geographical locations. In practice, however, there are plenty of masking techniques to replace real IP addresses, such as, for example, various VPN and proxy services. In these circumstances, it becomes important to enable more sophisticated functionality of extracting network information, possibly taking into account multiple factors. As a starting point, we can consider extracting information about the mobile network operator (provided that the user is on a mobile connection). This is quite similar to how Google differentiates Google Play users according to the country/region, and provides a country-specific version of the app marketplace, such that some apps are restricted to be downloaded and installed in some specific countries (e.g., mobile banking apps are typically limited to their specific countries, and are not expected to be used by foreigners from abroad).

- *Potential Integration with Other Existing Access Control Policy Languages*: as it was described, the current implementation is based on SANTA – an Interval Temporal

Logic-based policy language. In the future, however, we might consider integration the proposed approach with other existing languages and frameworks. Since the concepts of access control location and location-awareness a quite generic, it is expected to be applicable to a wide range of existing access control solutions. Experimenting with other languages will potentially prove the generic applicability of the proposed approach, or, otherwise, provide some insights on how it should be improved in this respect.

References

[1] Abadi, M., Burrows, M., Lampson, B., and Plotkin, G. (1993). A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 15(4):706–734.

[2] Abadi, M. and Fournet, C. (2003). Access control based on execution history. In *Proceedings of the Network and Distributed System Security Symposium (NDSS 2003)*, volume 3, pages 107–121.

[3] Ahn, G.-J. (2009). Discretionary access control. In *Encyclopedia of Database Systems*, pages 864–866. Springer.

[4] Allen, J. F. and Ferguson, G. (1994). Actions and events in interval temporal logic. *Journal of logic and computation*, 4(5):531–579.

[5] Almehmadi, A. and El-Khatib, K. (2013). Authorized! access denied, unauthorized! access granted. In *Proceedings of the 6th International Conference on Security of Information and Networks*, pages 363–367. ACM.

[6] Almehmadi, A. and El-Khatib, K. (2017). On the possibility of insider threat prevention using intent-based access control (ibac). *IEEE Systems Journal*, 11(2):373–384.

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

[8] AlZain, M. A., Pardede, E., Soh, B., and Thom, J. A. (2012). Cloud computing security: from single to multi-clouds. In *2012 45th Hawaii International Conference on System Science (HICSS)*, pages 5490–5499. IEEE.

[9] Anderson, A. (2005). A Comparison of Two Privacy Policy Languages: EPAL and XACML. Technical Report SMLI TR-2005-147, Sun Microsystems, Inc., Mountain View, CA, USA.

[10] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., Stoica, I., et al. (2009). Above the clouds: A Berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, University of California, Berkeley.

[11] Bakshi, K. (2009). Cisco cloud computing-data center strategy, architecture, and solutions. Technical report, Cisco Systems, Inc.

[12] Barhamgi, M., Bandara, A. K., Yu, Y., Belhajjame, K., and Nuseibeh, B. (2016). Protecting privacy in the cloud: Current practices, future directions. *Computer*, 49(2):68–72.

[13] Bechmann, A. and Lomborg, S. (2014). *The Ubiquitous Internet: User and Industry Perspectives*, volume 25. Routledge.

[14] Bell, D. E. and La Padula, L. J. (1976). Secure computer system: Unified exposition and multics interpretation. Technical Report ESD-TR-75-306, The Mitre Corporation.

[15] Bender, D. (2012). Privacy and security issues in cloud computing. *The Computer & Internet Lawyer*, 29(10):1–15.

[16] Bertino, E., Bettini, C., Ferrari, E., and Samarati, P. (1996). A temporal access control mechanism for database systems. *IEEE Transactions on knowledge and data engineering*, 8(1):67–80.

[17] Bertino, E., Bettini, C., Ferrari, E., and Samarati, P. (1998). An access control model supporting periodicity constraints and temporal reasoning. *ACM Transactions on Database Systems (TODS)*, 23(3):231–285.

[18] Bishop, M. (2003). What is computer security? *IEEE Security & Privacy*, 99(1):67–69.

[19] Bonatti, P., De Capitani di Vimercati, S., and Samarati, P. (2002). An algebra for composing access control policies. *ACM Transactions on Information and System Security (TISSEC)*, 5(1):1–35.

[20] Brucker, A. D. and Petritsch, H. (2009). Extending access control models with break-glass. In *Proceedings of the 14th ACM symposium on Access control models and technologies*, pages 197–206. ACM.

[21] Buxmann, P., Hess, T., and Ruggaber, R. (2009). Internet of services. *Business & Information Systems Engineering*, 1(5):341–342.

[22] Buyya, R., Vecchiola, C., and Selvi, S. T. (2013). *Mastering cloud computing: foundations and applications programming*. Newnes.

[23] Carminati, B., Ferrari, E., and Perego, A. (2006). Rule-based access control for social networks. In *OTM Confederated International Conferences 'On the Move to Meaningful Internet Systems'*, pages 1734–1744. Springer.

[24] Cau, A. and Moszkowski, B. (2018). Interval Temporal Logic. http://www.antonio-cau.co.uk/ITL/index.html.

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

[26] Council, W. and Heineman, G. (2001). Component-based software engineering putting the pieces together. *Addison Weysley*.

[27] Damianou, N., Dulay, N., Lupu, E., and Sloman, M. (2001). The ponder policy specification language. In *Policies for Distributed Systems and Networks*, pages 18–38. Springer.

[28] Damianou, N., Dulay, N., Lupu, E. C., and Sloman, M. (2000). Ponder: A language for specifying security and management policies for distributed systems. Technical Report DoC 2000/1, Imperial College, Department of Computing.

[29] Damianou, N. C. et al. (2002). *A policy framework for management of distributed systems*. PhD thesis, Imperial College London (University of London).

[30] Dautov, R., Distefano, S., Bruneo, D., Longo, F., Merlino, G., and Puliafito, A. (2017). Pushing intelligence to the edge with a stream processing architecture. In *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 792–799. IEEE.

[31] Dautov, R., Distefano, S., and Buyya, R. (2019). Hierarchical data fusion for Smart Healthcare. *Journal of Big Data*, 6(1):19.

[32] Dautov, R., Kourtesis, D., Paraskakis, I., and Stannett, M. (2013). Addressing self-management in cloud platforms: a semantic sensor web approach. In *Proceedings of the 2013 international workshop on Hot topics in cloud services*, pages 11–18. ACM.

[33] Dautov, R., Paraskakis, I., and Kourtesis, D. (2012). An ontology-driven approach to self-management in cloud application platforms. In *Proceedings of the 7th South East European Doctoral Student Conference (DSC 2012)*, pages 539–550.

[34] Dautov, R., Paraskakis, I., and Stannett, M. (2014a). Towards a framework for monitoring cloud application platforms as sensor networks. *Cluster computing*, 17(4):1203–1213.

[35] Dautov, R., Paraskakis, I., and Stannett, M. (2014b). Utilising stream reasoning techniques to underpin an autonomous framework for cloud application platforms. *Journal of Cloud Computing*, 3(1):13.

[36] De Nicola, R., Ferrari, G., Loreti, M., and Pugliese, R. (2013a). A language-based approach to autonomic computing. In *Formal Methods for Components and Objects*, pages 25–48. Springer.

[37] De Nicola, R., Loreti, M., Pugliese, R., and Tiezzi, F. (2013b). Scel: a language for autonomic computing. Technical report, IMT, Institute for Advanced Studies Lucca, Italy.

[38] Dillon, T., Wu, C., and Chang, E. (2010). Cloud computing: issues and challenges. In *2010 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, pages 27–33. IEEE.

[39] Douglas, P. F. (1966). The challenge of the computer utility.

[40] Endo, P. T. and Sadok, D. F. H. (2010). Whois based geolocation: A strategy to geolocate internet hosts. In *2010 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, pages 408–413. IEEE.

[41] Erl, T. (2005). *Service-oriented architecture: concepts, technology, and design*. Pearson Education India.

[42] Feltus, C. (2014). *Aligning access rights to governance needs with the responsibility metamodel (ReMMo) in the frame of enterprise architecture*. PhD thesis, Faculty of Computer Science, University of Namur, Belgium, University of Namur.

[43] Feng, D.-G., Zhang, M., Zhang, Y., and Xu, Z. (2011). Study on cloud computing security. *Journal of Software*, 22(1):71–83.

[44] Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., and Chandramouli, R. (2001). Proposed nist standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):224–274.

[45] Ferreira, A., Chadwick, D., Farinha, P., Correia, R., Zao, G., Chilro, R., and Antunes, L. (2009). How to securely break into rbac: the btg-rbac model. In *Annual Computer Security Applications Conference (ACSAC'09)*, pages 23–31. IEEE.

[46] Garfinkel, S. (1999). *Architects of the information society: 35 years of the Laboratory for Computer Science at MIT*. MIT press.

[47] Godik, S. and Moses, T. (2002). Oasis extensible access control markup language (xacml). Technical report, OASIS Committee Secification.

[48] Holden, T. (2018). New McAfee Report Reveals Data in the Cloud More Exposed Than Organizations Think. https://www.mcafee.com/content/enterprise/en-in/about/newsroom/press-releases/press-release.html?news_id=20181029005552.

[49] Hu, L., Ying, S., Jia, X., and Zhao, K. (2009). Towards an approach of semantic access control for cloud computing. In *IEEE International Conference on Cloud Computing*, pages 145–156. Springer.

[50] Hu, V. C., Ferraiolo, D., Kuhn, R., Friedman, A. R., Lang, A. J., Cogdell, M. M., Schnitzer, A., Sandlin, K., Miller, R., and Scarfone, K. (2013). Guide to attribute based access control (abac) definition and considerations. Technical Report 162, NIST special publication.

[51] Jajodia, S., Samarati, P., Sapino, M. L., and Subrahmanian, V. (2001). Flexible support for multiple access control policies. *ACM Transactions on Database Systems (TODS)*, 26(2):214–260.

[52] Jajodia, S., Samarati, P., and Subrahmanian, V. (1997). A logical language for expressing authorizations. In *Proceedings of 1997 IEEE Symposium on Security and Privacy*, pages 31–42. IEEE.

[53] Janicke, H., Cau, A., Siewe, F., and Zedan, H. (2007). Deriving enforcement mechanisms from policies. In *2007 Eighth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'07)*, pages 161–172. IEEE.

[54] Janicke, H., Cau, A., Siewe, F., and Zedan, H. (2012). Dynamic access control policies: Specification and verification. *The Computer Journal*, 56(4):440–463.

[55] Janicke, H., Cau, A., Siewe, F., Zedan, H., and Jones, K. (2006). A compositional event & time-based policy model. In *2006 Seventh IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'06)*, pages 173–182. IEEE.

[56] Jin, X., Krishnan, R., and Sandhu, R. (2012). A unified attribute-based access control model covering dac, mac and rbac. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 41–55. Springer.

[57] Khan, M. A. (2016). A survey of security issues for cloud computing. *Journal of network and computer applications*, 71:11–29.

[58] Kundra, V. (2011). Federal cloud computing strategy. Technical report, White House [Chief Information Officers Council].

[59] Latham, D. C. (1986). Department of defense trusted computer system evaluation criteria. *Department of Defense*.

[60] Lorch, M., Proctor, S., Lepro, R., Kafura, D., and Shah, S. (2003). First experiences using xacml for access control in distributed systems. In *Proceedings of the 2003 ACM workshop on XML security*, pages 25–37. ACM.

[61] MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F., Metz, R., and Hamilton, B. A. (2006). Reference model for service oriented architecture 1.0. *OASIS standard*, 12:18.

[62] Mell, P. and Grance, T. (2011). The NIST definition of cloud computing. Technical Report SP 800-145, National Institute of Standards and Technology.

[63] Minch, R. P. (2004). Privacy issues in location-aware mobile devices. In *Proceedings of the 2004 37th Annual Hawaii International Conference on System Sciences*, pages 10–18. IEEE.

[64] Mohamed, A. (2009). A history of cloud computing. *Computer Weekly*, 27.

[65] Moreau, L., Bradshaw, J., Breedy, M., Bunch, L., Hayes, P., Johnson, M., Kulkarni, S., Lott, J., Suri, N., and Uszok, A. (2005). Behavioural specification of grid services with

the kaos policy language. In *2005 IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2005)*, volume 2, pages 816–823. IEEE.

[66] Moszkowski, B. (1985). A Temporal Logic for Multilevel Reasoning about Hardware. *Computer*, 18(2):10–19.

[67] Moszkowski, B. and Manna, Z. (1983). Reasoning in interval temporal logic. In *Workshop on Logic of Programs*, pages 371–382. Springer.

[68] Moszkowski, B. C. (1994). Some very compositional temporal properties. In *Proceedings of the IFIP TC2/WG2.1/WG2.2/WG2.3 Working Conference on Programming Concepts, Methods and Calculi (PROCOMET '94)*, pages 307–326. North-Holland Publishing Co.

[69] Osborn, S. (1997). Mandatory access control and role-based access control revisited. In *Proceedings of the second ACM workshop on Role-based access control*, pages 31–40. ACM.

[70] Osborn, S., Sandhu, R., and Munawer, Q. (2000). Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Transactions on Information and System Security (TISSEC)*, 3(2):85–106.

[71] Petcu, D. (2013). Multi-cloud: expectations and current approaches. In *Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds*, pages 1–6. ACM.

[72] Pettey, C. (2009). Preparing for the Digital Transformation Economy: The Tools Needed to Build a Digitally Native Enterprise. https://www.gartner.com/newsroom/id/1035013.

[73] Pugliese, R. and Tiezzi, F. (2012). Sacpl: a simple access control policy language. Technical report, University of Florence, Italy.

[74] Rasmusson, L. and Aslam, M. (2012). Protecting private data in the cloud. In *The 2nd International Conference on Cloud Computing and Services Science (CLOSER 2012)*.

[75] Rescher, N. and Garson, J. (1969). Topological logic. *The Journal of Symbolic Logic*, 33(4):537–548.

[76] Ristenpart, T., Tromer, E., Shacham, H., and Savage, S. (2009). Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 199–212. ACM.

[77] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W. E. (1991). *Object-oriented modeling and design*, volume 199. Prentice-Hall Englewood Cliffs, NJ.

[78] Sandhu, R. (1988). Transaction control expressions for separation of duties. In *1988 Fourth Aerospace Computer Security Applications Conference*, pages 282–286. IEEE.

[79] Sandhu, R., Ferraiolo, D., Kuhn, R., et al. (2000). The nist model for role-based access control: towards a unified standard. In *ACM workshop on Role-based access control*, volume 2000, pages 1–11.

[80] Sandhu, R. S. (1993). Lattice-based access control models. *Computer*, 26(11):9–19.

[81] Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. (1996). Role-based access control models. *Computer*, 29(2):38–47.

[82] Sandhu, R. S. and Samarati, P. (1994). Access control: principle and practice. *IEEE Communications Magazine*, 32(9):40–48.

[83] Schroth, C. and Janner, T. (2007). Web 2.0 and soa: Converging concepts enabling the internet of services. *IT Professional*, 9(3).

[84] Segerberg, K. (1980). A note on the logic of elsewhere. *Theoria*, 46(2-3):183–187.

[85] Siewe, F., Cau, A., and Zedan, H. (2003). A compositional framework for access control policies enforcement. In *Proceedings of the 2003 ACM workshop on Formal methods in security engineering*, pages 32–42. ACM.

[86] Smyth, P. (2009). Cloud computing a strategy guide for board level executives. Technical report, Kynetix Technology Group.

[87] So, K. (2011). Cloud computing security issues and challenges. *International Journal of Computer Networks*, 3(5):247–55.

[88] Sundararajan, S. and Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing. *Journal of network and computer applications*, 34(1):1–11.

[89] Takabi, H. and Joshi, J. B. (2012). Semantic-based policy management for cloud computing environments. *International Journal of Cloud Computing*, 1(2-3):119–144.

[90] Takabi, H., Joshi, J. B., and Ahn, G.-J. (2010). Security and privacy challenges in cloud computing environments. *IEEE Security & Privacy*, 8(6):24–31.

[91] Theoharidou, M., Papanikolaou, N., Pearson, S., and Gritzalis, D. (2013). Privacy risk, security, accountability in the cloud. In *2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom)*, volume 1, pages 177–184. IEEE.

[92] Twidle, K., Dulay, N., Lupu, E., and Sloman, M. (2009). Ponder2: A policy system for autonomous pervasive environments. In *Fifth International Conference on Autonomic and Autonomous Systems (ICAS'09)*, pages 330–335. IEEE.

[93] Uszok, A., Bradshaw, J., Jeffers, R., Suri, N., Hayes, P., Breedy, M., Bunch, L., Johnson, M., Kulkarni, S., and Lott, J. (2003). Kaos policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. In *Proceedings of 2003 IEEE 4th International Workshop on Policies for Distributed Systems and Networks (POLICY'03)*, pages 93–96. IEEE.

[94] Uszok, A., Bradshaw, J. M., Johnson, M., Jeffers, R., Tate, A., Dalton, J., and Aitken, S. (2004). Kaos policy management for semantic web services. *IEEE Intelligent Systems*, 19(4):32–41.

[95] Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. (2008). A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39(1):50–55.

[96] Von Wright, G. H. (1979). A modal logic of place. In *The Philosophy of Nicholas Rescher*, pages 65–73. Springer.

[97] Wei, Y. and Blake, M. B. (2010). Service-oriented computing and cloud computing: Challenges and opportunities. *IEEE Internet Computing*, 14(6):72–75.

[98] Woo, T. Y. and Lam, S. S. (1993). Authorization in distributed systems: A new approach. *Journal of Computer Security*, 2(2-3):107–136.

[99] Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18.