Cloud Orchestration at the Level of Application

Project Acronym: **COLA**

Project Number: **731574**

Programme: **Information and Communication Technologies**
Advanced Computing and Cloud Computing

Topic: **ICT-06-2016 Cloud Computing**

Call Identifier: **H2020-ICT-2016-1**
Funding Scheme: **Innovation Action**

Start date of project: 01/01/2017 Duration: 30 months

Deliverable:

D7.1 COLA security requirements

Due date of deliverable: 30/04/2017 Actual submission date: 27/04/17

WPL: SICS

Dissemination Level: PU

Version: 1.0

1 Table of Contents

2 List of Figures and Tables

Tables

3 Status and Change History

Table 1 Status Change History

Status:	**Name:**	**Date:**	**Signature:**
Draft:	N. Paladi and A. Michalas	13/04/2017	N. Paladi
Reviewed:	P. Kacsuk	21/04/2017	P. Kacsuk
Approved:	T. Kiss	27/04/2017	T. Kiss

Table 2 Document Change History

Version	**Date**	**Pages**	**Author**	**Modification**
V01	10/02	13	C. Gehrmann	First version with outline and introduction
V01	14/02	13	N. Paladi	Edited and updated initial version
V01	15/02	13	A. Michalas	Review and minor corrections
V02	27/02	14	A. Michalas	Introduction
V02	28/02	15	A. Michalas	Objectives, Relation with other WPs and Deliverables, Organization
V02	06/03	16	A. Michalas	Security Requirements Process
V02	07/03	19	A. Michalas	Security Components
V02	09/03	24	N. Paladi	Describe "*Other requirements"* Add *Cloud Compute, Network, Storage Security Requirements*
V03	16/03	25	A. Michalas	Identified more core security requirements
V03	20/03	27	A.Michalas	Described COLA architecture in a high level
V04	03/04	30	A.Michalas	Prioritization of Core Security Requirements
V04	03/04	42	N. Paladi	Extensive update of Use case analysis
V05	11/04	42	N. Paladi	Update use case overview, comprehensive editing.
V06	12/04	42	A. Michalas	Review and edit Chapter 6
V07	13/04	42	A. Michalas	Update Chapter 9
V08	13/04	43	N. Paladi	Pre-review updates, update table captions
V09	27/04	45	N. Paladi	Add list of figures and update document according to reviewer comments
V1.0	27/04	47	N. Paladi	Final update, ready for delivery

4 Acronyms

Table 3 List of acronyms

COLA	Cloud Orchestration at the Level of Application
UML	Unified Modelling Language
MiCADO	Microservices-based Cloud Application-level Dynamic Orchestrator
CM	Credential Manager
PM	Policy Manager
TTP	Trusted Third Party
SPEL	Security Policy Enforcement Layer
TPM	Trusted Platform Module
CSP	Cloud Service Provider
ADV	Malicious Adversary
MAC	Message Authentication Code
PRF	Pseudorandom Function
DoS	Denial of Service Attack
IaaS	Infrastructure-as-a-Service
PII	Personally Identifiable Information
TPM	Trusted Platform Module
SGX	Software Guard Extensions

5 Definitions

Attestation protocol: a cryptographic protocol involving a target, an attester, an appraiser, and possibly other principals serving as trust proxies. The purpose of an attestation protocol is to supply evidence that will be considered authoritative by the appraiser, while respecting privacy goals of the target (or its owner).

Control plane: router architecture hardware and software components for handling packets destined to the device itself as well as building and sending packets originated locally on the device.

Forwarding plane: router architecture hardware and software components responsible for receiving a packet on an incoming interface, performing a lookup to identify the packet's IP next hop and determine the best outgoing interface towards the destination, and forwarding the packet out through the appropriate outgoing interface.

Integrity measurement records (also *integrity measurements*): a list of hashes recording some sequence of events, such as e.g. installation of set of binaries or opening files for reading or writing. The list is expanded on each event and cannot be forged assuming that the underlying cryptographic primitives are secure.

6 Introduction

Cloud computing has progressed from a bold vision to massive deployments in various application domains. However, the complexity of technology underlying cloud computing introduces novel security risks and challenges. Threats and mitigation techniques for the existing cloud models have been under intensive scrutiny in recent years [5], [6], while the industry has invested in enhanced security solutions and issued best practice recommendations.

Until recently, large scale computing was available exclusively to organizations with deep pockets and an abundance of in-house expertise. Cloud computing has changed that to the point when any user with even basic technical skills can obtain access to vast computing resources at low cost. In the technology adoption lifecycle, cloud computing has now moved from an early adopters' stage to an early majority, where we typically see explosive deployments. Throughout the past few years, many users have started relying on cloud services without even knowing it. Major web mail providers utilize cloud technology; tablets and smartphones often default to automatically uploading user photos to cloud storage and social networks; finally, several large publishers and book sellers store content – accessible from low-power e-book readers – in cloud storage. In other words, the adoption of cloud computing has moved from careful interest to intensive experimentation and is now rapidly approaching a phase of near ubiquitous use.

One of well-documented benefits of cloud computing is its ability to supply a variable amount of resources (computational power, storage, network capacity), which can scale dynamically up and down. On the demand side, we can see applications that are likely to be formed of one or more services. Services can be either in-house developed or provided by external suppliers or open-source communities. Services could also be shared between applications.

Nowadays, more businesses start migrating their services to cloud-based systems. This, has constantly increased the use of cloud computing while the applications that are deployed have different requirements (i.e. different amount of resources are needed). As a result, cloud implementations incorporate a lot of "moving parts". This is supported by what is known as Cloud Orchestration. Cloud Orchestration allows users to quickly and easily deploy an infrastructure (and even manage its lifecycle) in a consistent, repeatable way using fully configurable profiles. However, many of the existing orchestration platforms are lacking in security while security teams are also not aware of the explicit risks that these systems can pose.

Like any new technology, cloud computing creates new opportunities and poses new risks. The fact that cloud computing involves the aggregation of computing power, and more importantly, information, has become a source of increasing concern. Users, providers, and government policy-makers are having many concerns about the current use and future evolutionary path of cloud computing. Performing a security/risk assessment of orchestration platforms and governance/usage will not only guide us during the design and development phases of COLA, but will also provide valuable insights to protocol designers that wish to build even better and more secure orchestration frameworks.

Responding to these risks, COLA will provide a novel security architecture allowing efficient and secure deployment of arbitrary applications as well as advanced security policy

management and enforcement on orchestration level. To this end, we will first collect the necessary security requirements that we need to consider when building such frameworks. The collection and identification of the security requirements will be mainly driven by the design and the specific needs of the Microservices-based Cloud Application-level Dynamic Orchestrator (MiCADO) framework that the whole project will be based on. This is particularly important since it is expected that the defined security requirements will directly affect the MiCADO framework.

6.1 Scope

This document describes and lists the COLA security requirements. The requirements are the foundation for the COLA security architecture design as well as the novel security functions developed in the project. The document focuses mainly on high-level security requirements. The purpose of the document is to provide the basis for the overall security design and functions supported in COLA. The purpose of the document *does not* include eliciting or defining detailed low-level security requirements. The detailed security design and implementations, that will be provided later in the project, will meet both the applicable requirements defined in this document as well as further, more detailed requirements, which will be provided as part of the detailed design. These will include additional security requirements not listed in this document. The final COLA security design choices will be evaluated against the requirements in this document by the end of the project.

6.2 Objectives

The primary objectives of this document are to:

- Define the security requirements methodology;
- Identify the security requirements that will drive the design and development of the COLA security components;
- Rank those security requirements based on the COLA use cases;
- Indicate possible state-of-the-art techniques that may be useful for the COLA framework.

6.3 Relation with other WPs and Deliverables

Because of the central importance of data security in the COLA work plan, the security requirements have been separated from other technical requirements. However, for the proper collection of the security requirements this document should be read in conjunction with D4.2 “Requirements gathering and performance benchmarking of microservices” where the collection and definition of functional and non-functional cloud infrastructure and access layer level requirements of typical MiCADO microservices will be defined. In addition to that, this deliverable will also provide valuable insights to the D8.1 “Business and technical requirements of COLA use-cases” where the requirements of ISVs/technology providers and end users towards MiCADO platform developers to enable the migration of the demonstration applications and their efficient usage by end-users will be presented. Finally, this deliverable will provide the foundations for designing the overall security architecture of MiCADO which will be presented in D7.2. “MiCADO security architecture specification”.

6.4 Organization

This document begins with an introduction that describes the current landscape for modern, cloud-based applications and motivates the need for the advanced security features that COLA will provide. Chapter 5 defines the methodology used to identify the COLA security requirements and to rank those requirements. Chapter 6 gives a high level overview of the COLA architecture and presents the security components that will be designed. Chapter 7 presents a list of considered use cases, threat analysis, as well as a list of high-level requirements. Chapter 8 provides a list of other requirements that needs to be considered while Chapter 9 provides a concrete table of all the identified requirements.

7 Security requirements process

The IT sector over the decades has developed a set of standards and best practices to ensure the availability and reliable operation of computing services. In the early days, these standards and best practices barely touched on malicious activities, misbehaving users and security threats. However, the relentless growth of cybercrime and the exposition of vulnerabilities and flaws of existing systems by malicious adversaries, has forced the IT sector to develop (or to adapt) standards and best practices to deal with the emerging threats in cloud-based environments.

There are several ways of dealing with collection of security requirements. If the target of the security requirements engineering process is a specific product or system, there is a need for usage of more formal process or even using special purpose security requirements languages such as Abuse cases [1] or UMLsec [2]. As the security requirements we derive in COLA at the initial phase are *not intended* to completely define the security requirements of a single product or even a single system, we do not think it is appropriate to use such formal process. However, it would be worth considering the different steps that typically are involved in a product oriented security requirements collecting engineering process [3]:

- Specification of high level functional requirements.
- Identification of the deployment environment.
- Identification of assets and resources.
- Valuation of assets and resources
- Identification of users.
- Identification of potential attackers.
- Identification of attacker's interest in the resources/assets.
- Identification of attacker's capabilities.
- Specification of use cases.
- Specification of misuse scenarios.
- Identification of potential threats.
- Identification of security goals (derived through discussion with the stakeholders).
- Specification of high level security requirements such as security mechanisms to be incorporated.
- Specification of security policy and constraints on the working of the software and/or systems derived by discussing with the stakeholders (e.g., access control policy).
- Specification of low level functional requirements.
- Definition of exit criteria depending on the security state of the requirement specifications.
- Requirements inspection to identify security errors.
- Risk analysis.
- Performing security assessment of the requirement specifications.
- Specification of low level security requirements to remove security errors.
- Cost/benefit analysis.
- Categorization and prioritization of low level security requirements.
  Inclusion of selected low level security requirements in the requirement specifications.

Performing a meaningful security analysis of any system or category of system requires a high-level functional description of the target system. Similarly, it is necessary to identify the

users of the system in order to understand their main needs and expectations, also with respect to security. Furthermore, it is impossible to derive any meaningful design driving security requirements without having a sound picture of the main security threats to the system and their associated risks and costs.

One common approach to dealing with attacks are usage of so-called "attack trees" [4]. Attack trees are most useful tools both to derive security requirements and to valuate associated risk and are very well suited for analysis of specific products or small systems. However, deriving the complete attack tree for such large and complex target system as addressed by COLA would be *very complex* and hard. Instead, we will identify main threats in connection to the target architecture by analysing the main assets and components in the system and discussing – on a high level – potential attacks on these components and assets. We are convinced that such simplified process is a better way to identify the most important risks. The detailed attack and risk analysis should be done when products are developed based on the COLA architecture.

In summary, we have used the following simplified process for the security requirements collection in COLA:

- High level description of the COLA architecture analysis
  - Identification of key functional components in the COLA architecture
  - Identification of security components
  - Threats analysis
  - Identification of major architecture driving security requirements
  - Identification of major functional security requirements
- Use cases analysis
  - Initial security requirements based on questionnaire to the COLA use case stakeholders
  - Threats analysis
  - Deepened security analysis of the COLA uses cases
  - Identification of complementing security services use cases
- Security requirements listing
  - Structuring of high level architecture and use case driven security requirements
  - Break down of selected high level architecture and use-case driven security requirements
  - Security requirements complementing analysis
  - Identification of security requirements priorities

7.1 Requirements Quality

The COLA security requirements have been identified and collected using "best practice" approach as described above. Rather than using a strict requirements engineering process, we have used a pragmatic approach where the requirements have been gathered using a combination of system analysis and use-cases analysis. This method has been selected as we think such process is the most efficient considering the COLA goals, i.e., offering a cloud orchestration framework with high security quality and security service offering. A product level system will require much more detailed security analysis and design. However, the main security goal of COLA is to identify core system security risks and provide a *security*

architecture that *allows* offering a high-level security product offering using the architecture, rather than directly offer such product. Hence, a detailed security analysis and design is less important for such a prototype level system.

The quality of the derived security requirements has mainly been guaranteed through thorough internal review processes.

8 COLA architecture analysis

This chapter gives an overview of the intended COLA draft architecture with a main focus on the security components. In addition to that, we will present the main security requirements that will be derived from the defined use-cases provided by several partners within the consortium.

8.1 Architecture overview

In this section, we present a high level overview of the architecture, including a review of the intended functionality as well as a typical use-case scenario.

The COLA project targets to provide orchestration at the level of cloud resources and microservices. This feature uniquely identifies the project among similar solutions in this area. The overall system will aggregate the performance advantage of microservices with the dynamic behaviour of clouds. Orchestration combined with performance and health monitoring for an automatic fault recovery and healing system is the main target in this part of the MiCADO architecture.

Besides the orchestration service discussed above, the MiCADO orchestration layer will contain a Monitoring microservice and an Optimization decision maker microservice.
The Microservices discovery and execution layer of MiCADO will primarily utilize existing microservices, such as Consul, Docker, and Swarm (however, other emerging tools, such as Nomad will also be considered and investigated).

Consul is an open-source service discovery tool that also includes health check and alerts functionality. **Docker** is a kernel namespace based lightweight virtualisation solution that enables running microservices in containers. **Swarm** is a clustering mechanism built on Docker that is aware of worker nodes and their current workload, and is able to allocate new containers to the node currently least used.

Furthermore, the MiCADO architecture will be enhanced with certain security components that will be running as micorservices and will provide the necessary guarantees to increase the trustworthiness of the overall system. To this end, as discussed in the next section, MiCADO will be complemented with the following three main security components:

1. A security policy enforcement layer;
2. A credential manager;
3. A policy manager.

These components will use several cryptographic techniques in order to ensure that the overall function of the system is secure while the exchanged and stored data are successfully protected by strictly and well-defined privacy-preserving mechanisms. Finally, the access between different data as well as between different services that might be offered by multiple cloud deployments will be controlled by a set of security policies that will have the ability to update in real time based on certain malicious behaviors that will be identified during a proper run of the system.

8.2 Key components

In this section we describe the main components based on the MiCADO architecture. The security components will be thoroughly described in Security components.

Cloud infrastructure layer
This layer contains cloud instances, which in turn run containers that execute actual microservices. One instance can run one or more containers. As we will describe later in the security requirements, these cloud instances will have to run under a trusted state.

Cloud Access Layer
In order to support multiple heterogeneous cloud infrastructures, the COLA project will utilize the CloudBroker platform at the cloud access layer. The advantage of using such generic access layer is that the results of the project can be prototyped and validated on multiple heterogeneous clouds. The COLA testbed will include both commercial cloud resources (CloudSigma and Amazon) and also private academic clouds based on widely used open-source cloud middleware (OpenStack and Open Nebula).

Cloud Interface Layer
This is a set of APIs that provides means to launch and shut down cloud instances. There can be one or more cloud interfaces to support multiple clouds. Either native interfaces of targeted clouds can be applied (e.g. EC2) or generic cloud access layers that provide access to multiple heterogeneous clouds.

The following layers are part of the *microservices orchestration layer*:

Application infrastructure definition layer
This forms the basis for creating a functional application infrastructure. At this level, software components and their requirements as well as their interconnectivity are defined. This layer does not contain any application-specific data. For example, to provide HTTP based services, this layer can define that to provide this functionality, a MySQL database, Apache HTTP server and Nginx proxy server are needed, and that Nginx needs connection to Apache, which in turn needs connection to MySQL. As the infrastructure is agnostic to the actual application using it, this definition can be shared with any application that requires such an environment.

Application layer
This layer contains actual application code and data to make an incarnation of a defined application infrastructure function in such a way that the desired functionality is achieved. For example, this layer could populate a database with initial data, and configure an HTTP server with both look and feel and application logic.

8.3 Security components

The scope of this section is to introduce the main security components that COLA architecture is based on. To this end, we will only provide a high-level description of the components that will help us to successfully identify the core security requirements. A detailed description of the security components will be presented in Deliverable 7.2 “Security Architecture Design”. Apart from describing the main functionality of each component as well as the containing subcomponents (also referred as internal components) we will also

describe the relations between them. This will help us in the following phases of the project , namely during the design of the security architecture, as well as during the development and deployment phases.

Before we proceed with the description of each component, we provide a high-level overview of the main components that will be part of the COLA security architecture (**Error! eference source not found.**). The intent of this overview is to provide readers with valuable insights in order to better understand the role and the main functionality of each underlined component.

Following common design principles, our architecture is based on a layered approach, where correlated functionalities are grouped into a common layer that provides simple interfaces towards other layers and components, thereby abstracting the internal design and structure. The described architecture focuses on the technology-independent components of COLA, which can be integrated in various application and deployment scenarios. Furthermore, as mentioned earlier, in this deliverable we only provide generic interfaces that allow for a flexible integration of existing cloud services with the functionality offered by COLA. This general description of the security components can be valuable for companies and organizations that wish to implement the COLA functionality in their own private clouds.

A core objective of the COLA project is to design and implement in MiCADO a set of security modules supporting the security vision outlined above.
To this end, the following tasks must be carried out:

- Design and formalize security properties of external cloud computing and storage resources;
- Integrate into MiCADO routines for verification of external computing resources properties by extending pervious results from research on verification of IaaS resources;
- Design a security policy framework and a policy language for MiCADO;
- Research and design principles for application level security classification of application components and their data.

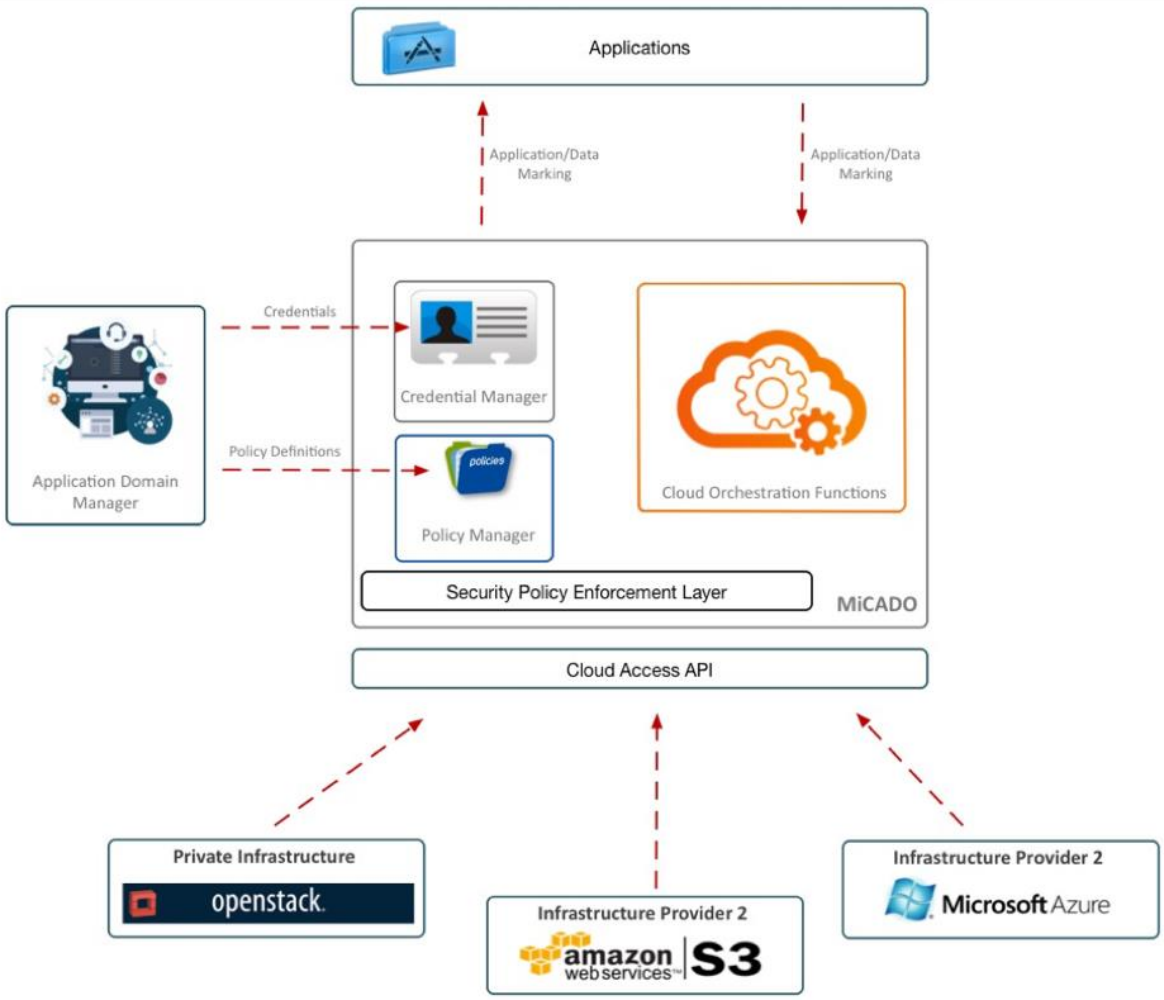

Figure 1 COLA Security Architecture

As illustrated in Figure 2, the COLA security architecture will be based on three main components. The Credential Manager (CM), the Policy Manager (PM) and the Security Policy Enforcement Layer (SPEL). In what follows, we briefly present each component and give a high level description of the operational functionality.

Credential Manager (CM)

The Credential manager is responsible for securely storing the credentials of entities that can access the MiCADO service. Credential manager can receive requests from any entity and is responsible for realising the corresponding credential in a secure and privacy-preserving way. In addition to that, all credentials that are managed by the CM should be stored such that the CM would be unable to reveal any valuable information about the content of the credential except the fact that there are valid. Hence, CM is not considered as a trusted entity. However, we explicitly assume that it will follow all protocol specification correctly.

Policy Manager (PM)

Policy manager provides a scalable way to manage the security of numerous applications and orchestration functions. More precisely, PM can define and distribute security policies, allow the installation of certain software to local or remote systems and monitor the activities of all

systems in the COLA framework to ensure compliance with corporate policies and centralized control. Having a centralized policy management helps users to use same access policies in multiple applications and CSPs. PM is communicating directly with the application domain manager. Hence, through certain monitoring processes it is easy to verify that the entire domain is protected as well as to modify the security settings when necessary.

Security Policy Enforcement Layer (SPEL)
The security policy enforcement layer in the MiCADO architecture will be responsible for the verification of both external and internal cloud resources for the application domain. Such verifications typically require interaction with external verification resources. For clarity, such connections are omitted in the overall architecture picture.

Trusted Third Party (TTP)
COLA uses a "trusted third party", with a key role in the overall framework and is trusted by the rest of the components. The role of TTP is of paramount importance for the security of COLA since it will be responsible for generating certain security guarantees about the trustworthiness of the cloud infrastructure that will be connected to MiCADO through the cloud access API. We rely on the commonly supported proposition that a large code base normally contains a proportionally large number of vulnerabilities. To reduce the code base, it is important that the TTP only supports the minimal necessary functionality. A TTP can communicate with components deployed on compute hosts to exchange integrity attestation information, authentication tokens and cryptographic keys. In addition to that, the TTP can verify the integrity of a pre-defined set of security-sensitive code and data executing or stored on the compute hosts. This can be done over an *attestation protocol* (see "Definitions") assuming that the hosts are equipped with common, commodity hardware-based isolation components or features – such as e.g. a Trusted Platform Module, or Software Guard Extensions [17]. In addition, the TTP can *seal* data to a the correct configuration of a compute host, such that the data is only available if certain pre-defined code and data on the host have not been modified. For the needs of COLA, TTP will be communicating with SPEL in order to provide certain information that is needed for the successful verification of a cloud resource. Finally, TTP can verify the authenticity of a client as well as perform necessary cryptographic operations.

8.4 Adversary Model

For the security needs of COLA, we will assume the adversary model that is described in N. Paladi, C. Gehrmann and A. Michalas. "Providing user security guarantees in public infrastructure clouds." *IEEE Transactions on Cloud Computing*. IEEE, 2016., which is based on the Dolev-Yao adversary model D. Dolev and A. C. Yao "On the security of public key protocols." *IEEE Transactions on Information Theory*, vol. 29, no. 2, 1983.. The adversary model builds on the consideration that a remote adversary (*ADV*) can use its privileged access to leak confidential information or perform unauthorized modifications to the configuration of the cloud deployment. This adversary model, along with additional details and explicit assumptions, is described below.

ADV – e.g. a rogue system administrator – can obtain remote access to any virtualization host maintained by the IaaS provider, modify its configuration and install arbitrary software. However, *ADV* cannot access, modify or forge *integrity measurement records* (see

"Definitions") containing the account of the installed software[1]. Such integrity measurement records can be stored using commodity hardware-based isolation components or features (such as e.g. a Trusted Platform Module, or Software Guard Extensions). Furthermore, *ADV* cannot access the configuration data and volatile memory of virtual machine instances or other computational tasks executing in isolated environments on the compute hosts of the IaaS provider (examples include Intel Software Guard Extensions [17] or AMD Secure Encrypted Virtualization [18] technologies). This property is based on the closed-box execution environment for guest VMs, as outlined in Terra [19] and further developed in [20] and [21]. We continue with a set of assumptions regarding the security of various components across the architectural layers or aspects of a cloud deployment.

Hardware and Software Integrity
Recent media revelations have raised the issue of hardware tampering en route to deployment sites G. Greenwald, "How the NSA tampers with US-made Internet routers," The Guardian, May 2014. andS. Goldberg, "Why is it taking so long to secure internet routing?," Communications of the ACM, vol. 57, no. 10, pp. 56–63, 2014.. We define tampering in this context as the deliberate altering or adulteration of a product, package, or system, often with the goal of inserting security vulnerabilities. We assume that the cloud provider has taken necessary technical and non-technical measures to prevent such hardware and software tampering. With respect to software, we assume that the integrity checksums of all software involved in the deployment and operation of the cloud deployment have been verified prior to usage.

Physical Security
We assume physical security of the data centers where the IaaS is deployed. This assumption holds both when the IaaS provider owns and manages the data center (as in the case of Amazon Web Services, Google Compute Engine, Microsoft Azure, etc.) and when the provider utilizes third party capacity, since physical security can be observed, enforced and verified through known best practices by audit organizations.
This assumption is important to build higher-level hardware and software security guarantees for the components of the IaaS.

Low-Level Software Stack
We assume that at installation time, the IaaS provider reliably records integrity measurements of the low-level software stack: the Core Root of Trust for measurement; BIOS and host extensions; host platform configuration; Option ROM code, configuration and data; Initial Platform Loader code and configuration; state transitions and wake events, and a minimal hypervisor. We assume the integrity measurement record is kept on protected storage with read-only access and the adversary cannot tamper with it.

Network Infrastructure
The IaaS network deployment is under physical and administrative control of the IaaS provider. However, the adversary can overhear, create, replay and drop arbitrary messages communicated between legitimate users and their resources (virtual machine instances, virtual routers and network functions, storage abstraction components).

Cryptographic Security: We assume both symmetric and asymmetric encryption schemes are semantically secure and the *ADV* cannot obtain the plain text of encrypted messages.

[1] Integrity Measurement Architecture: https://sourceforge.net/p/linux-ima/wiki/Home/#overview

We also assume the signature scheme is unforgeable, i.e. the *ADV* cannot forge the signature of any entity and that a Message Authentication Code (MAC) correctly verifies message integrity and authenticity. Furthermore, we assume that the *ADV*, with a high probability, cannot predict the output of a pseudorandom function (PRF). We explicitly exclude denial-of-service attacks (DoS) A. Michalas, N. Komninos, N. R. Prasad, and V. A. Oleshchuk, "New client puzzle approach for dos resistance in ad hoc networks," in Information Theory and Information Security (ICITIS), 2010 IEEE International Conference, pp. 568–573, IEEE, 2010. and focus on *ADV* that aim to compromise the confidentiality of data in the cloud environment offered by COLA.

8.5 Architecture driving requirements

Well-designed cloud orchestration relieves application developers from detailed management of the underlying computing resources. From a business perspective, this allows to cut operational costs by completely separating *application execution* from *application design* and by utilizing the most cost efficient available cloud infrastructure. From a security perspective however, protecting data and code in such cloud federations introduces significant challenges. While cloud orchestration *within* a private cloud infrastructure is rather straightforward, allowing secure orchestration across *multiple* cloud infrastructures – or in hybrid infrastructures consisting of a combination of private and public resources – is much more demanding. This is caused by the restrictions imposed by security considerations: while some application components or data objects can be executed or stored on private cloud infrastructure, they cannot be executed on a public cloud or will only be allowed to do so given that the properties and *security policies* applied on the public cloud are verified. Such verifications include (but are not limited to) the following properties:

- Authentication of cloud access APIs;
- Verification of computing resource properties (cloud platform attestation);
- Verification/enforcement of data encryption policies and properties;
- Verification of access control properties;

Consequently, a central role of a cloud orchestration layer that manages external computing resources is to offer the functionality for such verification. However, it is insufficient to simply create the basic framework that allows application APIs to support such functionality. Such a naïve approach obliterates the main vision of cloud service orchestration – namely to relieve developers from detailed cloud resource configuration and management. Instead, we need to build advanced security policy management mechanisms at the orchestration level, with the main aim to shield the developers from detailed security management. Ideally, the developer and/or the application domain owner should only provide general security policies as well as security credentials for the application domain. Similarly, application developers should only use high-level APIs to "mark" the security levels of applications and data handled by the applications. These inputs would then be used by the infrastructure definition and subsequently by a special purpose security policy enforcement layer to enforce the security policies at orchestration level. Figure 3 illustrates an overview of the main security components in connection to the overall COLA architecture and the integration with MiCADO.

Ensuring that cloud API calls are secure and allowing only approved services and communications over the offered APIs are among the basic security requirements in most cloud computing models. API security becomes more complex in dynamic federated cloud

environments where the orchestration layers must manage multiple credentials. Such security management services for federated clouds are currently lacking, and the COLA project plans to fill this gap by offering novel mechanisms built on the MiCADO functionality. API security enforcement will be implemented in the policy enforcement layer, while the corresponding policy management and credential management functions will be implemented in separate modules. Project members have conducted significant research into novel principles for cloud resource verification on platform level, transparent to end user applications. These results will be extended and integrated into the MiCADO architecture. In particular, this will allow application domain managers to require reliable integrity verification of the offered cloud platform resources as well as secure storage of permanently stored data on the external provider infrastructure.

8.6 COLA Main Security Requirements

In this section we present the main security requirements that will be considered in order to mitigate certain vulnerabilities and protect COLA from specific malicious behaviors. In Table 4, we present a concrete list of the identified security requirements. For the evaluation of each requirement we have followed the RFC 2119 S. Bradner(1997, March 1). Key words for use in RFCs to Indicate Requirement Levels. Retrieved May 30, 2015, from Internet Engineering Task Force (IETF): https://www.ietf.org/rfc/rfc2119.txt conventions. More precisely, we use the following list of words to identify the importance of each identified requirement. “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY” and “OPTIONAL”.

Table 4 Identified COLA Security Requirements

SR01	COLA SHOULD provide certain guarantees that the cloud providers that are connected to the service through the Cloud Access API are running under a trusted state (i.e. have been launched under a specific security profile).
SR02	COLA SHALL make sure that the cloud providers that are connected to the service through the Cloud Access API are using protecting users’ data from external attacks by encrypting the entire hard disks of the CSP.
SR03	COLA SHALL guarantee that the credentials of a user can be revoked without affecting the overall performance or the proper function of the service.
SR04	COLA should provide guarantees that all launched VM’s are running in a trusted state.
SR05	COLA SHOULD enforce SSL communication between all participating instances.
SR06	COLA SHOULD NOT be operational if SSL is terminated (e.g. recognize SSL-stripping techniques).
SR07	COLA SHOULD allow entities with certain access rights to define certain security profiles that will be considered as trusted.
SR08	COLA SHOULD use/propose a mechanism that protects sensitive data that are temporarily stored in the memory.
SR09	COLA SHOULD ensure that deployed applications in the Application Server are trusted using a mature trust model.

SR10	COLA SHALL be operational only if all security components are active.
SR11	COLA SHOULD provide a proper and efficient mechanism for key revocation of the misbehaving entities.
SR12	COLA SHALL use a key generation algorithm that guarantees that the generated keys are secure (i.e. long enough) and that secret keys are not statically stored in one place (e.g. in the application server or in the application).
SR13	COLA SHOULD guarantee that when a user's key is compromised the rest of the keys MUST not be revoked.

8.6.1 Cloud Compute Security Requirements (CCSR)

In this subsection we present the requirements towards secure provisioning of cloud network resources. The following requirements have been elicited based on a questionnaire completed by Outlandish LLC and Instrumentacion y Componentes S.A, as well as prior work [5][7][8][14].

- **CCSR-1:** *COLA SHOULD provide mechanisms to enforce secure destruction of workloads and configuration.*
  Rationale: Workloads that are not securely decommissioned may store sensitive data, which presents confidentiality risks if exposed.

- **CCSR-2:** *COLA SHOULD provide mechanisms for placement selection of workloads.*
  Rationale: Tenants may put forth requirements towards the placement of workloads according to pre-defined criteria, e.g. geographical, jurisdictional or administrative placement criteria, as well as according to the type and properties of the underlying virtualization host.

8.6.2 Cloud Network Security Requirements (CNSR)

In this subsection we present the requirements towards secure provisioning of cloud network resources. The following requirements have been elicited based on a questionnaire completed by Outlandish LLC and Instrumentacion y Componentes S.A, as well as prior work [10], [11], [12], [13].

- **CNSR-1:** *COLA SHOULD provide support access control for cloud network infrastructure.*

  Rationale: The cloud orchestrator should support deployment of mechanisms and access control model that can prevent an adversary from simultaneously gaining control over cloud network resources at all privilege levels and in all roles.

- **CNSR-2**: *COLA SHOULD provide support verification of deployed network configuration policies.*

  Rationale: Tenant network configuration policies submitted must be verified for compliance, using e.g. an integration verification engine, prior to deployment.

- **CNSR-3**: *COLA SHOULD provide support mechanisms for authentication,*

confidentiality and integrity protection of network infrastructure.

Rationale: All network communication on the cloud control and management planes must the authenticated as well as confidentiality and integrity protected.

- **CNSR-4**: *COLA SHOULD provide support traceability of network infrastructure management.*

  Rationale: A mechanism must be in place to offer traceability and non-repudiation for all configuration and network management commands and policies implemented on the cloud network infrastructure.

- **CNSR-5**: *COLA SHOULD provide support isolation of tenant policy domains.*

  Rationale: The cloud orchestrator should enforce strong network policy isolation, such that the effects of policies in a certain tenant domain have no effect on other domains.

- **CNSR-6**: *COLA SHALL verify submitted network configuration and management policies prior to deployment.*

  Rationale: New network management policies must run through prior to deployment, to minimize or exclude the effect of malicious policies on the network configuration.

- **CNSR-7:** *COLA SHALL enforce tenant quota isolation.*

  Rationale: A mechanism must be in place to ensure that the cloud orchestrator allocates resources according to the assigned quota.

- **CNSR-8:** *COLA SHALL support authentication of endpoints enrolled into the network infrastructure.*

  Rationale: Use of authentication policies for limiting endpoint access to network infrastructure is a core functionality in cloud deployments.

- **CNSR-9:** *COLA SHALL support deployment of secure communication channels for in-transit protection of data among the endpoints of the network infrastructure*.

  Rationale: In-transit protection of data is required to avoid data leakage while transferring data between cloud deployments.

- **CNSR-10:** *COLA SHALL support limiting access to the network infrastructure according to pre-defined network properties of the endpoints.*
  Rationale: Tenants may wish to configure network infrastructure to limit connectivity according to certain network properties, such as e.g. IP addresses.

8.6.3 Cloud Storage Security Requirements (CSSR)

In this subsection we present the requirements towards secure provisioning of cloud network resources. The following requirements have been elicited based on a questionnaire completed by *Outlandish LLC* and *Instrumentacion y Componentes S.A*, as well as prior work [5][6][7][14].

- **CSSR-1:** *COLA SHOULD provide mechanisms to enforce secure destruction of data upon storage decommissioning.*
  Rationale: Storage resources that are not securely decommissioned may contain sensitive data which presents confidentiality risks if exposed [15].

- **CSSR-2**: *COLA SHOULD provide mechanisms for placement selection of data.*
- **Rationale:** Tenants may put forth requirements towards the location of data storage according to pre-defined criteria, e.g. geographical, jurisdictional or administrative placement criteria, as well as according to the type and properties of the underlying storage hardware.

9 Use-case analysis

9.1 COLA use cases overview

The relevance of the MiCADO framework is decided by its applicability to solving the needs and addressing the requirements of its target users.

Describing the use cases and target application domains is a pre-requisite step in identifying the MiCADO application domain and aligning its functionality with the needs and projected applications by the end-users. In this process, the descriptions of application domain use cases have been elicited from five end-user organizations (further referred to as *verticals)*: *Outlandish*, *CloudSME* (combining the use cases of HKN and Rheinschafe GmbH), *Saker*, and *INY-SARGA*. The target organizations represent various service domains and business models, which contributes to describing a rich variety of use cases and viewpoints:

- HKN is a German Managed Hosting Company, focusing on building HA clusters for its customers. HKN's customers are normally small and medium sized, German companies.

- Rheinschafe GmbH from Duisburg, Germany is a Digital Agency founded with the main focus on developing websites with TYPO3 and digital communication.

- Outlandish is a 20-person cooperative digital agency specialising in middleware, usability, search and scalable data applications. Outlandish's main focus is on the interface between computers and users in insight-generation and data management. Outlandish have considerable experience building highly usable and intuitive data management solutions.

- Instrumentacion y Componentes S.A. provides high quality services and solutions with added value in IT and Communications, Energy, Laboratory Equipment, Electronics and Medical Equipment.

- Saker Solutions Limited has a mission to expand the benefits achieved from the use of simulation modelling. Saker a provider of simulation based tools, training, support and consultancy in the UK.

We continue with a detailed use case overview of the verticals. While detailed requirements of all COLA use-cases are collected in deliverable D8.1, a short overview of specific security requirements is given below.

9.1.1 Instrumentacion y Componentes S.A. (Inycom) Security Requirements

The security requirements are:

- End users need a user/password to access to the web interface.
- Option to transfer encrypted data in case personally identifiable information (PII) – such as citizens' data – kept by the Aragon Regional Government need to be processed in an external cloud. This option is not considered in the use case, but if somehow it should be needed this is how it has to be done.

- Databases will only be accessible from a restricted list of IPs (account for this when launching cloud instances).
- Data will only be stored in EU or associated countries with data protection regulations at least as restrictive as the EU one.

9.1.2 SAKER Security Requirements

The security requirements will vary based on the client. The system must be able to run on a private network or private cloud that is completely disconnected from the Internet to support any security concern. This will enable the system to be used by clients in government organisations where simulation applications involve data that is sensitive or restricted. The system also needs to run on public clouds to allow commercial clients to make use of public cloud resources.

File encryption may be seen as a requirement depending upon the client and application.

9.1.3 Outlandish Security Requirements

Outlandish employs a wide range of security technologies that should be potentially supportable by the MiCADO framework. Briefly, such security technologies include:

- Password management solution, (Zoho Vault);
- Ansible Vault is used to store other application secrets installed on machine images or managed instances in the cloud.
- Credentials for unlocking the Ansible Vault are secured with GPG to enhance auditability, wrapped in the suite of shell scripts Blackbox [28].
- Hashicorp's Vault [29], Biscuit [30], or Treehugger [31] application secret stores with better auditability are considered for future development.

Full disk encryption – typically unlocked at boot – may be necessary for particularly sensitive applications that store either PII or business critical information.

An explicit expectation is successful black box penetration testing of the deployment along with capability to be examined in detail by a white box security audit.

Support for Ansible roles to set up software requirements (for example NGINX and Node.js) is implicitly assumed.

9.1.4 CloudSME Security Requirements

The main security requirement put forth in this case is TLS support for end-users communicating with a front-end host.
Connection to within the cluster must be restricted to specific whitelisted IP addresses.

9.2 COLA use cases classification

Table 5 presents the use cases relevant for the MiCADO framework and identified based on a questionnaire submitted to the above organizations. The presented use cases have been

grouped into several *categories* in order to facilitate their further categorization and prioritization:

1. *Automation* – use case focus to automated deployment and instantiation of application workloads.
2. *Infrastructure* – use case focus on infrastructure planning, deployment and management. While this category is closely connected to category (1), it nevertheless operates on a different level and is an essential prerequisite to reliable and secure application functioning.
3. *Scalability* – use case focus on infrastructure and application bundle scaling according to the operational needs.
4. *Storage* – use case focus on provisioning and operation of secure and reliable storage for application needs.
5. *Application* – use case focus on supporting the application operational needs through secure and reliable resource orchestration.

Table 5 Use Case sources

Outlandish - UC_O; NKH-Rheinschafe - UC_R*; Saker - UC_S*; INY-SARGA - UC_I**. **Categories: *(1) Automation, (2) Infrastructure (3) Scalability; (4) Storage; (5) Application***

ID	Title	Cat.
UC_O1	Scalable hosting, testing and automation for SMEs and public sector organisations	(1)
UC_O2	Easy developer experience for present and future Outlandish projects	(5)
UC_O3	Generic hosting of the WordPress CMS for simple content-based websites	(2)
UC_O4	Generic hosting of our React, Node.js bridge to WordPress REST API	(2)
UC_O5	Dynamic scaling of a popular Composer repository for WordPress plugins and themes	(3)
UC_O6	Easy SAAS-style deployment of School Councils web app across UK schools	(1)
UC_O7	Deployment of the British Council's social media analysis tool to the German Goethe Institute	(5)
UC_R1	Planning, deployment and management of clusters	(2)
UC_S1	Bursting onto a private cloud from SakerGrid network	(3)
UC_S2	Bursting onto a public cloud from SakerGrid network	(3)
UC_S3	Bursting onto a suitably secure cloud from SakerGrid network	(3)
UC_I1	User database to collects data from citizens' interaction with the Aragon Government online applications	(4)
UC_I2	Recommendation system to improve citizens interaction with public services	(5)

The set of use cases outlined in Table 4 represent a broad and rich set of goals and resource requirements expressed by the verticals. While this list of relevant use cases and requirements towards an orchestration framework is still incomplete, it can serve as input to identifying and prioritizing the high-level security requirements towards the MiCADO framework and discussed below.

9.3 High level security requirements

In this section we revisit use cases defined by the verticals towards the MiCADO framework in order to elicit a focused set of high-level security requirements that will be addressed by

the implementation of the MiCADO framework. The target requirement set includes a subset of the *main security requirements* towards MiCADO, defined in Section 6.

- **R1**: *COLA SHALL enforce tenant quota isolation;*

This requirement is based on use cases UC_O6, UC_S1, UC_S2 described in Table 5 and corresponds to Cloud Network Security Requirement **CNSR-7.**

- **R2:** *COLA SHALL support limiting access to the network infrastructure according to pre-defined network properties of the endpoints;*

This requirement is based on use cases UC_O5, UC_O6, UC_I1 described in Table 5 and corresponds to Cloud Network Security Requirement **CNSR-10.**

- **R3:** *COLA SHOULD provide support access control for cloud network infrastructure;*

This requirement is based on use cases UC_O6, UC_O7, UC_R1, UC_S1, UC_S2, UC_S3 described in Table 5 and corresponds to Cloud Network Security Requirement **CNSR-1.**

- **R4:** *COLA SHOULD provide mechanisms for placement selection of data.*

This requirement is based on use cases UC_O3, UC_O4, UC_R1, UC_S1, UC_S2, UC_S3 described in Table 5 and corresponds to Cloud Storage Security Requirement **CSSR-2.**

- **R5:** *COLA SHOULD provide support traceability of network infrastructure management*

This requirement is based on use cases UC_O1-UC_O7, UC_S1, UC_S2, UC_S3, UC_R1, UC_I1, UC_I2 described in Table 5 and corresponds to Cloud Storage Security Requirement **CSSR-2.**

9.4 Security services use-cases

The security requirements identified above can be addressed by a combination of five abstract security services. To clarify such services, the tables below provide a short description of relevant use cases for each for the identified requirement-service pair.

ID	Service-R1
Title	Tenant quota isolation enforcement
Description	Deployment mechanisms provides high-entropy authentication keys that allow to reliably negotiate confidentiality and integrity protection mechanisms between the tenant and the workloads deployed on cloud infrastructure.
Primary Actor	Tenant
Preconditions	Cloud infrastructure up and running
Postcondition	Workloads deployed with strong authentication keys
Main success scenario	1. Tenant generates authentication credentials. 2. Tenant provisions workload images with authentication credentials. 3. Tenant uploads encryption workload images to cloud image store. 4. Deployment mechanism instantiates workloads on cloud

	infrastructure. 5. Tenant communicates with workloads over an authenticated, confidentiality and integrity protected channel.
Frequency of Use	At each deployment of workloads.

ID	Service-R2
Title	Workload node verifier
Description	Deployment mechanism verifies nodes according to a pre-defined security policy.
Primary Actor	Deployment mechanism
Preconditions	Cloud infrastructure up and running
Postcondition	A set of trusted nodes is identified for future workload placement.
Main success scenario	1. Deployment mechanism conducts periodic sanity checks of the nodes in the cloud infrastructure. 2. Nodes that successfully pass the verification step are whitelisted as trusted according to the respective policy. 3. Workloads requiring an execution environment compliant with a certain security policy are deployed only on compliant nodes.
Frequency of Use	Periodically, implementation-specific.

ID	Service-R3
Title	Access control mechanism
Description	Infrastructure access control enforcement mechanism verifies the access credentials of tenants aiming to communicate with a workload and only accepts authorized users.
Primary Actor	Infrastructure access control enforcement mechanism.
Preconditions	Workloads are deployed and executing on the infrastructure.
Postcondition	Authorized tenant gained access to the workload. Unauthorized tenant access rejected.
Main success scenario	1. Tenant initiates communication with the cloud infrastructure in order to gain access to a workload. 2. Communication is mediated by an access control mechanism. 3. Access control mechanism verifies tenant credentials with respect to the requested workload. 4. Tenant with valid credentials is allowed access to the requested workload.
Frequency of Use	At each tenant to workload access attempt.

ID	Service-R4
Title	Workload node selector
Description	Deployment mechanism instantiates workloads exclusively on nodes that satisfy a certain security or placement policy.
Primary Actor	Deployment mechanism
Preconditions	Infrastructure nodes have been verified by Service-R2
Postcondition	Workload has been placed on a verified node that satisfies the

	workload placement policy.
Main success scenario	1. Deployment mechanism receives as input a workload with a deployment policy. 2. Deployment mechanism identifies infrastructure nodes that satisfy the deployment policy according to latest verification data. 3. Deployment mechanism deploys workload on one of the nodes satisfying the placement policy.
Frequency of Use	At each deployment of a workload with placement policy.

ID	Service-R5
Title	Interaction monitor
Description	Interaction monitor reliably records and reports events in the infrastructure (interaction of tenants with network, compute and storage resources) according to a configuration and a security policy.
Primary Actor	Interaction monitor
Preconditions	Workloads have been deployed on the infrastructure.
Postcondition	Tenant-workload interactions have been recorded and reported.
Main success scenario	1. Tenant deploys workload on an infrastructure node. 2. Adversary interacts with workload with malicious purposes. 3. Interaction monitor records interactions and raises an alarm. 4. Administrator acts upon the alarm, interaction monitoring data is used for forensic investigations.
Frequency of Use	At each interaction between components of the cloud infrastructure.

The security service use cases – along with the following threat analysis – serve as input towards the security architecture of the MiCADO framework developed in the COLA project.

9.5 Threat analysis

A threat analysis must start with a thorough threat taxonomy and identification in each specific context. A threat has a potential to exploit vulnerabilities and harm assets. Threat identification can be made based on history of previous incidents (if it exists) or an external threat catalogue. The approach adopted in this document has been to perform the identification of relevant threats through an assessment of a subset of use cases reported above. All use cases are evaluated regarding the possible threats in the context of cloud resource orchestration. The advantage of this approach if compared to the one based on a predefined list of threats is that it can allow one to address the 'known unknown' or the 'unknown unknown' threats and therefore it allows for identification of individual threats depending on the specific context.
For this reason a set of the COLA use cases defined in above has been used to drive the threat analysis. The use cases have been analysed to gain an understanding of:

- the main threat/s the use case is exposed to
- the vulnerability exploited by the threat (threat's description)
- the category the threat belongs to
- the impact caused by the threats
- the assets the attacker would be interested in

- the entry point where a potential attacker could interact with the service and/or business-model described in the use case
- possible mitigation that is the set of controls or measures that could prevent the threat from causing impacts

The threat analysis based on COLA use cases is carried out using a clearly defined structure, to ensure that the correct information has been collected. For this purpose a specific template has been defined to derive threat descriptions from COLA use cases and facilitate the risk analysis associated with each threat. The template is illustrated in Table 6:

Table 6 Threat analysis template

ID: Unique ID # of the threat	*Numbering scheme: <T_UC-number_associated-threat-number>,* *e.g.* T_R1_1, T_R1_2, T_R2_1, …
Name: Brief name of the threat	
Description: Detailed description of threat and its importance	
Category: ITU-T X.805 security dimension(s) – tick the appropriate box(es)	☐ Access control ☐ Authentication ☐ Non-repudiation ☐ Data confidentiality ☐ Communication security ☐ Data integrity ☐ Availability ☐ Privacy
Potential effect: What global effect it will have on major cloud domains (network, hosts, applications, e2e effect…)	
Assets impacted: What assets could be damaged?	☐ Data Assets: ☐ *Data* ☐ *Metadata* ☐ Computation Assets: ☐ *Software* ☐ *Hardware* ☐ *Configuration* ☐ Network Assets: ☐ *Software* ☐ *Data* ☐ *Configuration* ☐ Service provider IT Infrastructure: ☐ *IT Infrastructure* ☐ *Billing systems* ☐ *Operator data* ☐ *End user data*

	☐ Cloud provider physical infrastructure: ☐ *Facilities* ☐ *Energy Power* ☐ Human agents: ☐ *Cloud Service Operators* ☐ *End User Application Developers* ☐ *End User Application Administrators* ☐ *End User Service Providers* ☐ *End Users* ☐ Others (please specify): ☐ ☐
Possible Mitigation Hints (optional, if foreseen): How can we protect against the threat?	
Entry Points (optional, if known): What possible means does an adversary have?	

Based on earlier threat model and selected requirements, we continue with a detailed description of threats related to each of the identified security requirements. Note that the list of threats is not exhaustive, and other threats may be discovered as the security landscape evolves.

ID: Unique ID # of the threat	**T_R1_1**
Name: Brief name of the threat	Unauthorized access to a cloud domain
Description: Detailed description of threat and its importance	Isolation of the domain may fail allowing tenant to gain an access to resources belonging to the operator or other domains. This may jeopardize availability and security of the tenants and other cloud providers' services.
Category: ITU-T X.805 security dimension(s)	☒ Access control; ☐ Authentication; ☐ Non-repudiation; ☒ Data confidentiality; ☒ Communication security; ☒ Data integrity; ☒ Availability; ☐ Privacy
Potential effect: What global effect it will have on major cloud domains (network, hosts, applications, e2e effect…)	Availability and security of operators' resources and service provider's resources jeopardized. This may prevent opportunities that are gained by cloud services to multiple tenants.
Assets impacted: What assets could be damaged?	☒ Data Assets: ☐ *Data* ☐ *Metadata*

	☒ Computation Assets: ☐ *Software* ☐ *Hardware* ☐ *Configuration* ☒ Network Assets: ☐ *Software* ☐ *Data* ☐ *Configuration* ☐ Service provider IT Infrastructure: ☐ *IT Infrastructure* ☒ *Billing systems* ☐ *Operator data* ☒ *End user data* ☐ Cloud provider physical infrastructure: ☐ *Facilities* ☐ *Energy Power* ☐ Human agents: ☐ *Cloud Service Operators* ☐ *End User Application Developers* ☐ *End User Application Administrators* ☐ *End User Service Providers* ☐ *End Users*
Possible Mitigation Hints (if known): How can we protect against the threat?	Strong isolation between domains is needed. Authentication and authorization over the access to cloud management plane. Security monitoring is needed to detect ongoing incidents.
Entry Points (if known): What possible means does an adversary have?	Failing or misconfigured authentication and authorization both in the control plane or forwarding plane may enable access to domains.

ID: Unique ID # of the threat	**T_R2_2**
Name: Brief name of the threat	Compromised workloads
Description: Detailed description of threat and its importance	The user database that processes personally identifiable information (or other sensitive data) could be compromised if placed on an insecure cloud infrastructure.
Category: ITU-T X.805 security dimension(s)	☒ Access control; ☐ Authentication; ☐ Non-repudiation; ☒ Data confidentiality; ☒ Communication security; ☒ Data integrity; ☒ Availability; ☒ Privacy
Potential effect: What global effect it will have on major cloud domains (network, hosts, applications, e2e effect…)	Confidentiality, integrity and availability of e2e communication and workloads are compromised.
Assets impacted:	☒ Data Assets:

What assets could be damaged?	☐ *Data* ☐ *Metadata* ☒ Computation Assets: ☐ *Software* ☐ *Hardware* ☐ *Configuration* ☒ Network Assets: ☐ *Software* ☐ *Data* ☐ *Configuration* ☐ Service provider IT Infrastructure: ☐ *IT Infrastructure* ☐ *Billing systems* ☐ *Operator data* ☐ *End user data* ☐ Cloud provider physical infrastructure: ☐ *Facilities* ☐ *Energy Power* ☐ Human agents: ☐ *Cloud Service Operators* ☐ *End User Application Developers* ☐ *End User Application Administrators* ☐ *End User Service Providers* ☐ *End Users*
Possible Mitigation Hints (if known): How can we protect against the threat?	Applying security verification procedures – technical and organisational - for assuring that the hosts executing workloads are trustworthy. Only authenticated and authorized entities should be allowed to add nodes. Security monitoring of behaviour of added nodes as well as communication over the network.
Entry Points (if known): What possible means does an adversary have?	Software, image used for deploying new nodes may be compromised. Forwarding logic may be misconfigured so that illegitimate node, switch is able to get access to data flows. In this case, the malicious node is unintentionally added to the core network.

ID: Unique ID # of the threat	**T_R3_1**
Name: Brief name of the threat	Unauthorized access to data or workloads in a cloud
Description: Detailed description of threat and its importance	A failure of the deployment and configuration tool may allow unauthorized third parties to gain access to sensitive data and wokload resources. This may jeopardize availability and security of data and workloads.
Category: ITU-T X.805 security dimension(s)	☒ Access control; ☒ Authentication; ☐ Non-repudiation; ☐ Data confidentiality; ☒ Communication security; ☒ Data integrity; ☒ Availability; ☐ Privacy
Potential effect: What global effect it will have on major cloud domains (network, hosts,	Availability and security of operators' resources and service provider's resources jeopardized. This may prevent opportunities that are gained by cloud services to multiple tenants.

applications, e2e effect…)	
Assets impacted: What assets could be damaged?	☒ Data Assets: ☐ *Data* ☐ *Metadata* ☒ Computation Assets: ☐ *Software* ☐ *Hardware* ☐ *Configuration* ☒ Network Assets: ☐ *Software* ☐ *Data* ☐ *Configuration* ☐ Service provider IT Infrastructure: ☐ *IT Infrastructure* ☒ *Billing systems* ☐ *Operator data* ☒ *End user data* ☐ Cloud provider physical infrastructure: ☐ *Facilities* ☐ *Energy Power* ☐ Human agents: ☐ *Cloud Service Operators* ☐ *End User Application Developers* ☐ *End User Application Administrators* ☐ *End User Service Providers* ☐ *End Users*
Possible Mitigation Hints (if known): How can we protect against the threat?	Strong isolation between domains is needed. Authentication and authorization over the access to cloud management plane. Security monitoring is needed to detect ongoing incidents.
Entry Points (if known): What possible means does an adversary have?	Failing or misconfigured authentication and authorization both in the control plane or forwarding plane may enable access to domains.

ID: Unique ID # of the threat	**T_ R4_1**
Name: Brief name of the threat	Compromised data
Description: Detailed description of threat and its importance	The user database that contains personally identifiable information (or other sensitive data) could be compromised if placed on an insecure cloud infrastructure.
Category: ITU-T X.805 security dimension(s)	☐ Access control ☐ Authentication ☐ Non-repudiation ☒ Data confidentiality ☒ Communication security ☒ Data integrity ☐ Availability ☐ Privacy

Potential effect: What global effect it will have on major cloud domains (network, hosts, applications, e2e effect…)	A data store placed on a compromised cloud host presents a threat not only to the integrity of the data itself, but also to the entire tenant deployment. Middleware often has very weak inter-node defence mechanisms. Thus, one compromised node can be used as an attack vector to compromise the entire deployment.
Assets impacted: What assets could be damaged?	☒ Data Assets: ☐ *Data* ☐ *Metadata* ☒ Computation Assets: ☐ *Software* ☐ *Hardware* ☐ *Configuration* ☒ Network Assets: ☐ *Software* ☐ *Data* ☐ *Configuration* ☐ Service provider IT Infrastructure: ☐ *IT Infrastructure* ☐ *Billing systems* ☐ *Operator data* ☐ *End user data* ☐ Cloud provider physical infrastructure: ☐ *Facilities* ☐ *Energy Power* ☐ Human agents: ☐ *Cloud Service Operators* ☐ *End User Application Developers* ☐ *End User Application Administrators* ☐ *End User Service Providers* ☐ *End Users*
Possible Mitigation Hints (optional, if foreseen): How can we protect against the threat?	In order to protect against this threat, the cloud orchestrator should deploy tasks only on cloud hosts that belong to the same security domain. The solution may include remote attestation protocols and investigation in statistics data processing.
Entry Points (optional, if known): What possible means does an adversary have?	An adversary can have one or all the following means: Communication channels, user equipment and a network component

ID: Unique ID # of the threat	**T_ R5_1**
Name: Brief name of the threat	Undetectable unauthorized actions
Description: Detailed description of threat and its importance	An adversary may eliminate all traces of a comprimise in order to prevent detection and remediation activities.

Category: ITU-T X.805 security dimension(s)	☐ Access control ☐ Authentication ☒ Non-repudiation ☐ Data confidentiality ☐ Communication security ☒ Data integrity ☐ Availability ☐ Privacy
Potential effect: What global effect it will have on major cloud domains (network, hosts, applications, e2e effect…)	Lack of monitoring and audit data can prevent detection of unauthorized actions in a cloud deployment. Furthermore, in case such unauthorized actions have been detected, lack of monitoring or incorrect audit information can hinder, prevent or mislead investigative actions.
Assets impacted: What assets could be damaged?	☒ Data Assets: ☐ *Data* ☐ *Metadata* ☐ Computation Assets: ☐ *Software* ☐ *Hardware* ☒ *Configuration* ☐ Network Assets: ☐ *Software* ☐ *Data* ☒ *Configuration* ☐ Service provider IT Infrastructure: ☐ *IT Infrastructure* ☐ *Billing systems* ☒ *Operator data* ☒ *End user data* ☐ Cloud provider physical infrastructure: ☐ *Facilities* ☐ *Energy Power* ☐ Human agents: ☐ *Cloud Service Operators* ☐ *End User Application Developers* ☐ *End User Application Administrators* ☐ *End User Service Providers* ☐ *End Users*
Possible Mitigation Hints (optional, if foreseen): How can we protect against the threat?	In order to prevent stealthy adversary activities in the cloud deployment, it is important to configure the deployment with reliable and resilient auditing and monitoring mechanisms.
Entry Points (optional, if known): What possible means does an adversary have?	An adversary can have one or all the following means: Communication channels, user equipment and a network component

The threats described above can be addressed by implementing reliable mitigation mechanisms. While a variety of disparate tools and mechanisms are currently available, they

are poorly integrated with the state of the art cloud deployment and orchestration mechanisms. The MiCADO framework can incorporate tools and mechanisms that would effectively address and prevent the identified threats.

10 Other requirements

The requirements in this section have been elicited based on a requirements questionnaire completed by *Outlandish LLC* and *Instrumentacion y Componentes S.A.* The other participating verticals did not express any requirements that fall into this category.

10.6 Usability requirements

- **UR-1** *Configuration language used by the cloud orchestrator should allow dependencies to be managed between multiple types of resources.*
  Rationale: Dependency management is essential for a secure and well configured deployment. Missing or misconfigured dependencies can lead to serious security vulnerabilities.

- **UR-2** *Configuration language used by the cloud orchestrator should be implementation-agnostic*
  Rationale: It is important that the configuration language used by the cloud orchestrator is expressive enough to describe deployment details across multiple platforms.

- **UR-3** *Template should encourage intuitive exploration. Template engine should be explainable through examples.*
  Rationale: Poor usability of systems and tools is a very common source of human-induced errors. Therefore, it is essential to facilitate the use of the template engine for developers.

- **UR-4** *The template used for workload configuration must be plain text, human readable for an initiated developer*
  Rationale: Same as for UR3.

10.7 Software requirements

- **SwR-1** *Orchestrator platform should support various forms of isolation and containerization.*
  Rationale: Operating system level virtualization (e.g. containers) along with hardware virtualization (e.g. virtual machines) are among the most common mechanisms to ensure process and memory isolation in deployments. It is essential that the orchestration platform provides support for deployment and management of workloads using such mechanisms.

10.8 Data requirements

- **DR-1** *Orchestrator platform should support dynamic and secure creation and reliable destruction of data storage.*
  Rationale: Rapid provisioning of data storage capacity is among the core functionality of cloud deployments. The cloud orchestrator must support provisioning of storage fulfilling the requested security criteria. It equally important to ensure secure and reliable *deletion* of allocated data storage.

10.9 Performance requirements

- **PR-1** *Orchestrator platform should implement elasticity through secure dynamic creation, migration and destruction of workloads, as well as provisioning of network, compute and storage capacity on demand.*
  Rationale: Rapid provisioning of compute, network and storage resources is among the core functionality of cloud deployments. The cloud orchestrator must support provisioning of resources fulfilling the requested security criteria. Furthermore, such rapid provisioning must not impose performance penalties on other tenants.

11 Identified requirements and priorities

As a prelude to the full set of use case scenarios that will be used to validate COLA against the security and other technical requirements, each of the partners bringing use cases to the project ranked the importance of the identified security requirements on a scale of 0 – 10. Table 5 shows the ranking by each of the partners, ordered by the average.

Even though all partners that provide use-cases to the project participated in this ranking process, SAKER did not because they lack an understanding of the implications behind the various points. SAKER are not software security or hardware security experts and therefore we decided that it is not safe to attempt to prioritize a list of requirements that they do not understand in depth. To this end, we had several meetings in which we tried to identify the security requirements based on the services that will be offered by SAKER.

To provide some context for the evaluation, the general use cases for each partner are:

- **OUTLANDISH:** Collection, processing and storage of event sales and survey information. This may include collection and processing of personally identifiable information, requiring protection of application secrets and data at rest.
- **INYCOM:** Collection, storage and processing of potentially personally identifying information of users interacting with the government of Aragon Region. Collected information is processed either on-premise or using external infrastructure within the EU.
- **SAKER:** Computer simulation of processes for a variety of private and public customers. Simulations are conducted either on-premise or using external infrastructure. Such external infrastructure may range from public clouds to private and air-gapped deployments, presenting a variety of security requirements.
- **CloudSME:** website development, deployment and operation of high-availability cloud infrastructure clusters using commodity software.

Each of partners listed above contributes to the project an existing production application to verify that the software is useful in industrial contexts. As stated earlier, these will be expanded to a full set of detailed use case scenarios in a future document.

As can be seen from Table 5 the two most highly ranked requirements (rank 1) concern the security of the communication channels between the all entities that will be participating in the COLA scenarios. More precisely, all use-case partners ranked the existence of SSL-enabled channels as top priority in order to avoid possible man-in-the-middle attacks.

The next highly ranked group (ranks 2 and 3) contains requirements about both the generation and storage of secret keys. Based on the collected rankings, the generation of cryptographically secure keys is of paramount importance since such keys will be used for encryption and protection of sensitive data. In addition to that, key revocation of misbehaving users will play a key role for the overall security of COLA. Hence, during the design of COLA architecture we will need to build revocation mechanisms that will allow certain entities (e.g. administrator, data owner etc.) to revoke access for possible misbehaving users in a secure and *efficient* way. Efficiency here is considered as very important. Even though

there are many revocation mechanisms, providing a revocation process in an efficient way is considered as a difficult and emerging problem Antonis Michalas. “Sharing in the Rain: Secure and Efficient Data Sharing for the Cloud”. Proceedings of the 11th IEEE International Conference for Internet Technology and Secured Transactions (ICITST’16), Barcelona, Spain, December 5-7, 2016., Antonis Michalas and Noam Weingarten. “HealthShare: Using Attribute-Based Encryption for Secure Data Sharing Between Multiple Clouds”. Proceedings of the 30th IEEE International Symposium on Computer-Based Medical Systems (CBMS’17), Thessaloniki, Greece, 2017. .

The remaining requirements touch upon a variety of different advanced security features. There is more variation in the rankings of these requirements by the individual partners, reflecting differences in the project’s use cases. Nonetheless as can be seen from Table 5, the average importance of all of the identified security requirements is relative high, indicating that they are all directly relevant to the use cases and to the project.

Table 7 Rating of Security Requirements by Use Cases

Rank	Code	Description	OUTLANDISH	INYCOM	CloudSME
1	SR05	COLA SHOULD enforce SSL communication between all participating instances.	5	10	10
1	SR06	COLA SHOULD NOT be operational if SSL is terminated (e.g. recognize SSL-stripping techniques).	5	10	10
2	SR11	COLA SHOULD provide a proper and efficient mechanism for key revocation of the misbehaving entities.	5	9	8
3	SR12	COLA SHALL use a key generation algorithm that guarantees that the generated keys are secure (i.e. long enough) and that secret keys are not statically stored in one place (e.g. in the application server or in the application).	5	6	10
4	SR08	COLA SHOULD use/propose a mechanism that protects sensitive data that are temporarily stored in the memory.	2	8	10
4	SR13	COLA SHOULD guarantee that when a user’s key is	5	5	10

		compromised the rest of the keys MUST not be revoked.			
5	SR02	COLA SHALL that the cloud providers that are connected to the service through the Cloud Access API are using protecting users' data from external attacks by encrypting the entire hard disks of the CSP.	3	7	8
5	SR04	COLA should provide guarantees that all launched VM's are running in a trusted state.	3	7	8
6	SR01	COLA SHOULD provide certain guarantees that the cloud providers that are connected to the service through the Cloud Access API are running under a trusted state (i.e. have been launched under a specific security profile).	3	9	5
7	SR03	COLA SHALL guarantee that the credentials of a user can be revoked without affecting the overall performance or the proper function of the service.	4	8	3
7	SR10	COLA SHALL be operational only if all security components are active.	4	8	3
8	SR09	COLA SHOULD ensure that deployed applications in the Application Server are trusted using a mature trust model.	3	6	5
9	SR07	COLA SHOULD allow entities with certain access rights to define certain security profiles that will be considered as trusted.	3	5	5

12 References

[1] J. McDermott and C. Fox, “Using Abuse Case Models for Security Requirements Analysis,” *In Proceedings of the 15th Computer Security Applications Conference (ACSAC'99)*, Pheonix, AZ, USA, IEEE CS Press, 1999, pp. 55-64.

[2] J. Juerjens, Secure Systems Development with UML, Springer, 2005.

[3] M. U. A. Khan and M. Zulkernine, “On selecting appropriate development processes and requirements engineering methods for secure software,” *in Proc. 33rd Annual IEEE International Computer Software and Applications Conference*, Seattle, WA, 2009, pp. 353-358.

[4] B. Schneier, Secrets and lies: digital security in a networked world. Wiley, 2004.

[5] A. Michalas and K. Y. Yigzaw, "LocLess: Do you Really Care Where Your Cloud Files Are?". *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Luxembourg City, 2016, pp. 515-520, 12-15 Dec, 2016.

[6] A. Michalas, "Sharing in the rain: Secure and efficient data sharing for the Cloud," *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*, pp. 182-187, 5-7 Dec, Barcelona, Spain, 2016.

[7] N. Paladi and A. Michalas, "“One of our hosts in another country”: Challenges of data geolocation in cloud storage". *2014 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE)*, pp. 1-6, Aalborg, Denmark, 11-14 May, 2014.

[8] A. Michalas, N. Paladi and C. Gehrmann, "Security aspects of e-Health systems migration to the cloud," *2014 IEEE 16th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pp. 212-218. Natal, Brazil, 15-18 Oct, 2014.

[9] S. Bradner(1997, March 1). Key words for use in RFCs to Indicate Requirement Levels. Retrieved May 30, 2015, from Internet Engineering Task Force (IETF): https://www.ietf.org/rfc/rfc2119.txt

[10] N. Paladi and C. Gehrmann. "Towards Secure Multi-tenant Virtualized Networks." *Trustcom/BigDataSE/ISPA, 2015 IEEE*. Vol. 1. IEEE, 2015.

[11] Nicolae Paladi. "Towards secure SDN policy management." *Utility and Cloud Computing (UCC), 2015 IEEE/ACM 8th International Conference on*. IEEE, 2015.

[12] N. Paladi and C. Gehrmann. "TruSDN: Bootstrapping Trust in Cloud Network Infrastructure." *12th EAI International Conference on Security and Privacy in Communication Networks*. 2016.

[13] E. Haleplidis, K. Patras, K. Pentikousis, S. Denazis, J. Hadi, S. Mojatatu, M. D., and O. Koufopavlou, “Forwarding and control element separation (ForCES) Protocol Specification,” tech. rep., RFC 7426, 2015

[14] N. Paladi, C. Gehrmann and A. Michalas. "Providing user security guarantees in public infrastructure clouds." *IEEE Transactions on Cloud Computing*. IEEE, 2016.

[15] Jordon, Michael. "Cleaning up dirty disks in the cloud." *Network Security* 2012.10 (2012): 12-15.

[16] D. Dolev and A. C. Yao ”On the security of public key protocols.” *IEEE Transactions on Information Theory*, vol. 29, no. 2, 1983.

[17] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, “Innovative Instructions and Software Model for Isolated Execution,” in *Proceedings of the 2Nd International Workshop on Hardware and Architectural Support for Security and Privacy*, HASP ’13, (New York, NY, USA), pp. 10:1–10:1, ACM, 2013.

[18] T. Lendacky, "x86: Secure Memory Encryption (AMD)." https://lwn.net/Articles/ 685215/. Accessed: 2016-11-20.

[19] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh, "Terra: A virtual machine-based platform for trusted computing," *in ACM SIGOPS Operating Systems Review*, vol. 37, ACM, 2003.

[20] A. Seshadri, M. Luk, N. Qu, and A. Perrig, "SecVisor: A tiny hypervisor to provide lifetime kernel code integrity for commodity OSes," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 6, 2007.

[21] F. Zhang, J. Chen, H. Chen, and B. Zang, "Cloudvisor: retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization," *in Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pp. 203–216, ACM, 2011.

[22] G. Greenwald, "How the NSA tampers with US-made Internet routers," The Guardian, May 2014.

[23] S. Goldberg, "Why is it taking so long to secure internet routing?," Communications of the ACM, vol. 57, no. 10, pp. 56–63, 2014.

[24] A. Michalas, N. Komninos, N. R. Prasad, and V. A. Oleshchuk, "New client puzzle approach for dos resistance in ad hoc networks," in Information Theory and Information Security (ICITIS), 2010 IEEE International Conference, pp. 568–573, IEEE, 2010.

[25] A. Michalas, N. Komninos, and N. R. Prasad, "Mitigate dos and ddos attack in mobile ad hoc networks," International Journal of Digital Crime and Forensics (IJDCF), vol. 3, no. 1, pp. 14–36, 2011.

[26] A. Michalas, N. Komninos, and N. Prasad, "Multiplayer game for ddos attacks resilience in ad hoc networks," in Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology (Wireless VITAE), 2011 2nd International Conference on, pp. 1–5, Feb 2011.

[27] A. Michalas, N. Komninos, and N. R. Prasad, "Cryptographic puzzles and game theory against dos and ddos attacks in networks," International Journal of Computer Research, vol. 19, no. 1, p. 79, 2012.

[28] Blackbox https://github.com/StackExchange/blackbox

[29] Hashicorp Vault https://www.vaultproject.io/

[30] Biscuit https://github.com/dcoker/biscuit

[31] TreeHugger https://github.com/timeoutdigital/treehugger

[32] Antonis Michalas. "Sharing in the Rain: Secure and Efficient Data Sharing for the Cloud". Proceedings of the 11th IEEE International Conference for Internet Technology and Secured Transactions (ICITST'16), Barcelona, Spain, December 5-7, 2016.

[33] Antonis Michalas and Noam Weingarten. "HealthShare: Using Attribute-Based Encryption for Secure Data Sharing Between Multiple Clouds". Proceedings of the 30th IEEE International Symposium on Computer-Based Medical Systems (CBMS'17), Thessaloniki, Greece, 2017.