CLOUD ACCOUNTABILITY PROJECT

D:C-8.1 Framework of evidence

Deliverable Number:	D38.1
Work Package:	WP 38
Version:	Final
Deliverable Lead Organisation:	UiS
Dissemination Level:	PU
Contractual Date of Delivery (release):	30th June, 2014
Date of Delivery:	23rd June, 2014

Editor
Tomasz Wiktor Wlodarczyk (UiS) Rui Pais (UiS)

Contributors
Bikash Agrawal, Håvard Molland, Hafiz Gulzar (UiS), Thomas Rübsamen, Christoph Reich (HFU), Monir Azraoui, Melek Önen (EURECOM), Tobias Pulls (KAU), Jean-Claude Royer (ARMINES-EMN)

Reviewers
Siani Pearson (HP) Brian Dzminski (QMUL) Carmen Fernandez (UMA)

Table of Contents

Index of Figures

Index of Tables

Executive Summary

Evidence is one of the key elements of an accountable system. It provides the basis for users' trust and as a result influences the likelihood that users would be willing to use the system. The evidence framework consists of a set of mechanisms for extracting evidence in typical scenarios encountered in cloud services. The approach is focused on continuous monitoring of predefined activities, recording necessary data, and further analysis to complete the extraction process. These steps build upon rules and obligations defined in WP C-4 and B-3, and the conceptual architecture defined in WP C-2. Outcomes of the framework of evidence are then utilized, e.g., to elicit metrics in WP C-5. The current document is an initial report, scheduled to M21 and will be updated at M30.

The guiding principles for the A4Cloud Framework of Evidence are:
1. Enabling third party audits with privacy preservation (to support auditors and government entities)
2. Light weight design (to enable efficient implementation)
3. Fixating the logs and other elements of evidence (to support non-repudiation and trust)
4. Following location of data and performed operations (to support attributability)

Basing on the existing literature and work performed in the project we have defined **Accountability Evidence** as a collection of data, metadata, routine information and formal operations performed on data and metadata, which provide *attributable* and *verifiable* account of the fulfilment of relevant obligations with respect to the service and that can be used to convince a third party of the veracious (or not) functioning of an *observable* system.

The Framework of Evidence follows a scheme of five general steps:
1. Plan ahead the monitoring of events that relate to accountability and supporting sources;
2. Collect a predefined description of events, and map demonstrative elements that provide support;
3. Assemble the elements in an evidence record, referencing the supporting elements;
4. Timestamp the records by a trusted service, guaranteeing its origin and integrity;
5. Securely save the record within the cloud service provider.

The event's records shall constitute accountability evidence, and may be made available to related stakeholders (as Data Subjects relatively to their data processing), auditors and regulators. The elements in the system providing the support of the observed actions will not be stored with the record, but merely referred by location and a cryptographic digest, avoiding large amounts of duplicate data. The linked reference between records is made with aggregation of a timestamp, identification of the entity compiling the record, and the hash of the record itself. A secret salt (known only by the service provider) may also be included in the hashing process, intended to compromise brute-force attacks attempting to determine the records' contents. The linking process is complete by including the previous reference in the chain.

An evidence analyser performs validation relating and comparing elements from both records and policies. Elements expressed in the policies define the agreed obligations, authorizations and access rules. The appropriate values present in the evidence will be considered, and the actions and operations matched against corresponding policy obligations.

The framework of evidence relies on the existence of monitor and log components in cloud systems, and in A4Cloud tools as the Automated Audit System, AAS, as the main extraction mechanism of evidence elements.

1 Introduction

A growing interest in developing accountability for distributed computing, and particularly for services and business practices in the cloud services, introduces new approaches to control and monitoring of digital systems.

The work in accountability of digital systems addresses several major trends in the cloud and other Internet services. There is an increasing quantity of data captured, stored and processed to provide additional services. Users become increasingly dependent on cloud services where they have little to no control over data processing routines, sometimes with and sometimes without being aware of consequences and solutions to the problem. Accountability of cloud systems becomes a critical prerequisite to ensure an acceptable level of control over data in such services through the combination of socio-economic, legal, regulatory and technical approaches (Pearson et al. 2012). It provides the basis for users' trust and as a result influences the likelihood that users would be willing to use the system.

Three types of mechanisms are necessary to achieve accountability: preventive, detective and corrective. Evidence is one of the key elements of an accountable system, central to the detective mechanism and auxiliary to preventive and corrective mechanisms.

Accountability is a planned process. Evidence collection is defined upfront and based on a carefully designed framework. This way the evidence should be available at any time or stage of the service delivery process. This allows for a much wider spectrum of actions and shorter reaction time. The evidence can be used to ascertain that policies were complied with, or vice versa, to show that they were not. From this perspective it can be argued that for accountability of cloud services, evidence becomes equally important as a fundamental piece in any preventive and corrective mechanisms to implement. These considerations also lead to an account-oriented definition of evidence, what can be designated as accountability evidence.

In the A4Cloud Conceptual Framework, three different levels of verification are suggested, which are organisational policies, IT controls and mechanisms, and operations (see Figure 1, adapted from WP C-2). Evidence for the accountability of services provides elements and support to the different layers of the services' composition. As it is suggested, necessarily elements must be gathered that enable verification of:

- policies compliance, according to the definition of the services and agreed policies,
- information about the mechanisms and procedures implemented, and
- continuously operational information, with a dominant emphasis in the detection of privacy and security breaches.

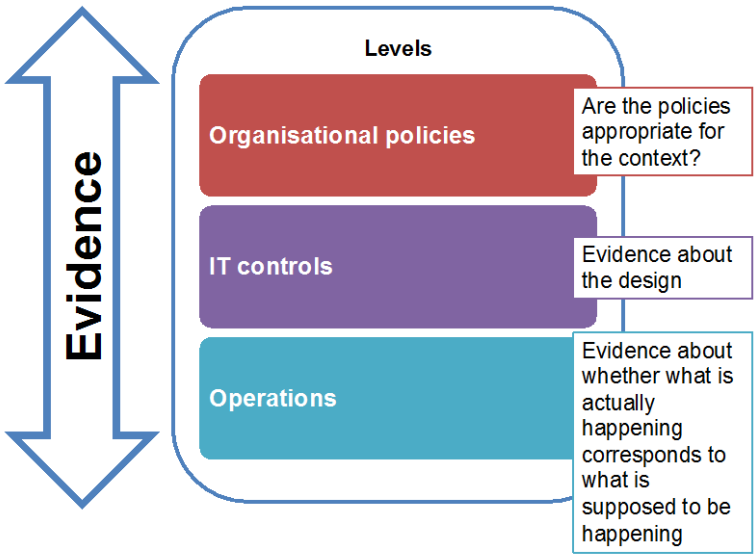


Figure 1: Evidence provision for verification at different service levels

Is important to note that accountability does not replace or directly contain digital forensics; however there is an important body of knowledge available in the forensics field particularly to help defining convincing and trustworthy evidence elements for particular scenarios. Forensics can further enhance evidence in the case of detection of non-compliance with policies in serious or contested cases (after the event has already happened and additional evidence is required).

The evidence framework consists of a set of mechanisms for extracting evidence in typical scenarios encountered in cloud services. The approach is focused on continuous monitoring of predefined activities, in special those that reflect the services' need of accountability provision, recording necessary data, and further analysis to complete the extraction process. These steps build upon rules and obligations defined in WP C-4 and B-3, and conceptual architecture defined in WP C-2. Outcomes of the framework of evidence are then utilized, e.g., to elicit metrics in WP C-5.

It is of course impossible to cover all possible scenarios. Accordingly, the typical scenarios included in this deliverable are based on use cases developed by WP B-3 and analysis performed earlier in WP C-8 in tasks T:C-8.1 and T:C-8.2. Scenarios are used to identify possible sources of evidence. This may have included real-time logging data, information about system configuration, particularly concerning security mechanisms (access control, key management, among others), certification and seals demonstrating that specific measures and practices are in place (e.g., updated security software and hardware solutions, staff's training certificates), etc.

This document is organized as follows. Section 2, Evidence for Accountability, relates the role of evidence with the different aspects of accountability considered in the A4Cloud project, and we propose a definition for accountability evidence. Section 3, Requirements of Framework of Evidence, details the requirements considered for the proposed framework of evidence, including a brief summary of identified requirements. Section 4, Framework of Evidence, presents the proposed framework with detailed sections for each major component. Section 5, Auditing Process and Evidence, describes the role of the auditing system and its automation by audit agents. Section 6, Business Use Case: Health Care, considers obligations detected in the general context of an e-Health Care cloud service from a A4Cloud proposed use case, as example of application. Conclusions and future work are set in Section 7. Some of the formal aspects of the framework of evidence, such as a demonstration of tamper-evident evidence repository, are addressed in the Appendix, A1.

1.1 Related Works

We can identify two general types of related work, one related with evidence in digital systems and the other with secure logging and auditing systems. The first type of related work is not directly focused on evidence provided or collected to support accountability of cloud services. It relates primarily with forensics, where digital and other evidence are collected *in loco* to attempt possible attributions, or demonstrate malicious or criminal practices or intents. However, many ideas can be applied or at least provide inspiration for accountability evidence.

There exist several overview papers dealing with digital, computer or electronic forensics. While differences exist between the particular areas of forensics they do not appear to be significant enough to prevent to establish a general definition for digital evidence. The overview works by Stamm, Wu, and Liu 2013, Volonino 2003 and Dixon 2005 mention the evidence term extensively in certain practical problems, but do not attempt to discuss its general definition.

The widely cited book "Digital Evidence and Computer Crime" (Casey 2011) provides a definition of digital evidence as "*any data stored or transmitted using a computer that support or refute a theory of how an offense occurred or that address critical elements of the offense such as intent or alibi.*" This definition stems directly from traditional forensics and as such from legal tradition and low of evidence, based on earlier work by Chisum 2011. Further search for definition of evidence in digital forensics context did not return significantly different results.

At the moment we are not aware of any work on a formal definition of evidence for accountability for the Cloud or other Internet service. There are works available in social, economic and political accountability that link to evidence; however, they typically are not of direct relation to the accountability as defined for digital systems. Important exceptions to this are works on accountability and evidence in elections, particularly the transition from paper to electronic elections. (Mercuri 2002) showed basing on 2000 US election that many new voting products provide less accountability than traditional methods. Verifiability of such voting protocols was analysed by (Kremer, Ryan, and Smyth 2010). However, discussing evidence definition has not been attempted in these works.

1.1.1 Related works on digital evidence for accountability

In "A survey of accountability in computer networks and distributed systems", (Zhifeng Xiao, Nandhakumar Kathiresshan 2012), the authors present a generic review of accountability characteristics and elements to be provided from different contexts in computer networking and distributed environments. Accountability in Cloud computing is described as the ability of the system to provide: "*Identities*: every event definitely linked to the system that execute it. *Secure record*: the machine keeps a note of past events such as systems (…). *Auditing*: The record can be examined for traces of errors. *Evidence*: When an auditor notices an error, it can attain proof of the error that can be confirmed separately by a moderator." Properties for audits (completeness, accuracy, verifiability) and for the logs (tamper-evident, time-stamping) are also considered and analysed, with the proposition that "tamper-evident logs can offer a firm foundation for accountable clouds". Further analysis includes: Accountable Virtual Machines, collaborative monitoring, accountable MapReduce and verifiable resource accounting.

The work "Accountability as a Service for the Cloud", (Yao et al. 2010b), is one of the few directly focused on evidence provision for accountability of cloud services, and the closest we found based on similar considerations we have taken in this deliverable; conceptualization of a framework of evidence for accountability of cloud services, with non-disputable logging based on monitoring and auditing. For that reason we detail this work in greater depth. The authors introduce their concept of accountability, which includes attribution as a major task, and with the core functionality: logging, monitoring and auditing, and dispute resolution.

The proposed approach consists of a novel design to achieve Trustworthy Service Oriented Architecture (TSOA), to identify and associate failures and misbehaviours with the responsible entities, administrated by a new service, designated Accountability as a Service (AS). The AS service controls the accountability functions, which are kept separate from the operational service domain.

Evidence logging is formalized with the definition of evidence event, which is consequently wrapped in its associated meta-information (like timestamp, event description, etc.) forming what is designated as a Trace. Each actor in an interaction of services keeps local copies of the traces and a hashed and

encrypted version, named token, is kept by the AS. The logging procedure: logging, authorizing invoicing and execution, is designed to achieve strong accountability. System monitoring and auditing includes a logic mechanism that verifies compliance based on analysis of pre-established SLAs and business operation logic that defines the correct flow between services. Policies definition are introduced to fill the gap between SLA and business logic definitions and monitoring mechanisms used to evaluate service legitimacy, and where services are responsible for defining evidence semantics: what should be provided in the tokens, which elements should be used and how they can be used to verify compliance. The work includes tests performed on an evaluation system implemented on Amazon EC2.

In the work of (Wang and Zhou 2010), a collaborative monitoring mechanism is proposed for making multitenant platforms accountable. The proposition considers a third party external service to provide a "supporting evidence collection", containing evidence for SLAs compliance checking defined distinctively from run-time logs). This type of service is presented as *Accountability services*, offering "a mechanism for clients to authenticate the correctness of the data and the execution of their business logic in a multitenant platform". The external accountable service contains a Merkle B-tree structure with the hashes of the operation signatures concatenated with the new values of data after occurrence of state changes. The work includes algorithms for logging and request processes, and an evaluation of a testing environment implemented in Amazon EC2.

Yumerefendi and Chase, 2007, propose a network storage service with strong accountability, which annotates operations with evidence of correct execution, offering audit and challenge interfaces to enable clients to verify the server. The service, designated CATS, relies on asymmetric cryptography and an "external publishing medium" where each actor of the network storage periodically publishes a digest of its state. Experiments with a CATS prototype were used to evaluate the cost of accountability under several conditions, with results showing that strong accountability is practical in mission-critical distributed services with strong identity.

Digital objects need endless maintenance due to their dependency on hardware, software and continually changing technology standards every few years; hence dependency on *digital records* is a concern with the mount of formal e-business. How digital records can be protected, utilized, proved and comprehended over time is conditional to the legal, administrative and technological contexts where they will be taken under consideration. The work "Long-term trusted preservation service using service interaction protocol and evidence records" (Blažič, Klobučar, and Jerman 2007) presents an *approach and a solution* to tackle the problem related to long-term integrity, authenticity and validity provision of digital data as evidence. The suggested system introduces a standard Trusted Archive System (TAS), based on a digital evidence standard, *Evidence Record Syntax* (Brandner, Pordesch, and Gondrom 2014). The TAS is established on *PKI-enabled* infrastructures for trust creation, and long-term data integrity proofs, and a service interaction protocol: Long-term Archive Protocol. Some other works of reference on digital evidence are the seminal work on storage formats are Digital Evidence Bags (DEB) (Turner 2005) and the extension ontology based approach to correlate event log based evidence, "An open architecture for digital evidence integration" (Schatz and Clark 2006), using semantic web technologies for describing and representing event log based digital evidence.

Some related papers focusing on the notion of evidence analysis for auditing and compliance checking are: (Vaughan et al. 2008; Guts, Fournet, and Nardelli 2009; Le Metayer, Mazza, and Potet 2010), and (Mazza, Potet, and Le Métayer 2011).

In the work from (Guts, Fournet, and Nardelli 2009), we find the following definition: "a protocol is auditable with respect to a property if it logs enough evidence to convince an impartial third party, called a judge, of that property." And then type systems are proposed, which allows to statically verifying that a protocol logs enough evidences.

The work from (Vaughan et al. 2008) proposes rich authorization logic based on a dependently-typed system. They introduce the language Aura that uses a notion of reference monitor to automatically log accesses to a resource. It also checks the access and generates a proof of the access for auditing. This is a kind of proof carrying approach; proof elements are first class values and are manipulated by the language. The type system is proved to verify subject-reduction and normalization properties.

The notion of evidence is also at the centre of the work of (Le Metayer, Mazza, and Potet 2010). The authors argue the following: "we believe that the means to constitute evidence that could be used in case of conflict should be considered from the onset of IT projects and be part of the requirements for the design of IT systems." They specify a framework for log architectures, with actions from malicious or secure agents and they define criteria to characterize acceptable architectures. This work brings the notion of claims that can be correctly and precisely evaluated from the logs. The paper focuses on the actions of malicious agents and how to characterize log architectures to get consistent logs and to disproving erroneous claims.

A related work from (Mazza, Potet, and Le Métayer 2011) defines a formal framework for specifying and reasoning about logs as electronic evidence. More precisely, they consider decentralized, distributed logs, and an analysis method, defined prior to legal disputes, to determine liability of the parties for predefined misbehaviours. The framework presented allows “to specify liability, claims, and logs as electronic evidence.”. Claims are defined a priori, with attached properties for a given claim, avoiding the need of general properties describing the behaviour of the system. The model uses the B-method, a methodology that allows specifications focused both on data and behaviours, referenced as a standard in industry. This formal framework is based on agents participating in some forms of interactions, described as events, and exchanging messages representing those interactions. Origin, authentication, and integrity of messages are assumed by hypotheses; however, it is claimed that the framework can easily be adapted for working without such assumptions.

2 Evidence for Accountability

2.1 Accountability in the A4Cloud Project

In this section we briefly enumerate the different elements proceeding from other works in the A4Cloud project that constituted source of insight for this report, for reference and contextualization. We mainly considered accountability attributes and practices identified in the WP C-2, the conceptual framework, the WP B-3 provided the use-case development, actors, roles and identified obligations, and from previous work in this WP C-8, the identification of types of evidence. Then we will proceed by proposing a definition for accountability evidence.

2.1.1 Accountability attributes

Accountability attributes, described in the A4Cloud Conceptual Framework (Felici and Pearson 2013), encompass concepts that are considered part of and supporting accountability. At the time of writing seven attributes of accountability are considered:

- **Observability** is a property of an object, process or system which describes how well the internal actions of the system can be described by observing the external outputs of the system.
- **Verifiability** is a property of an object, process or system that its behaviour can be verified against a requirement or set of requirements.
- **Attributability** is a property of an observation that discloses or can be assigned to actions of a particular actor (or system element).
- **Transparency** is a property of an accountable system that it is capable of ‘giving account’ of, or providing visibility of, how it conforms to its governing rules and commitments.
- **Responsibility** is the state of being assigned to take action to ensure conformity to a particular set of policies or rules.
- **Liability** is the state of being legally obligated or responsible.
- **Remediability** is the state (of a system) of being able to correct faults or deficiencies in the implementation of a particular set of policies and rules and/or providing a remedy to a party, if any, harmed by the deficiency.

These attributes have different importance from the perspective of the framework of evidence. From A4Cloud Conceptual Framework, accountability attributes are classified into two logical groups as shown in Figure 2, those that reflect on how such concept should or could be implemented and those that reflect on the accountability as a concept.

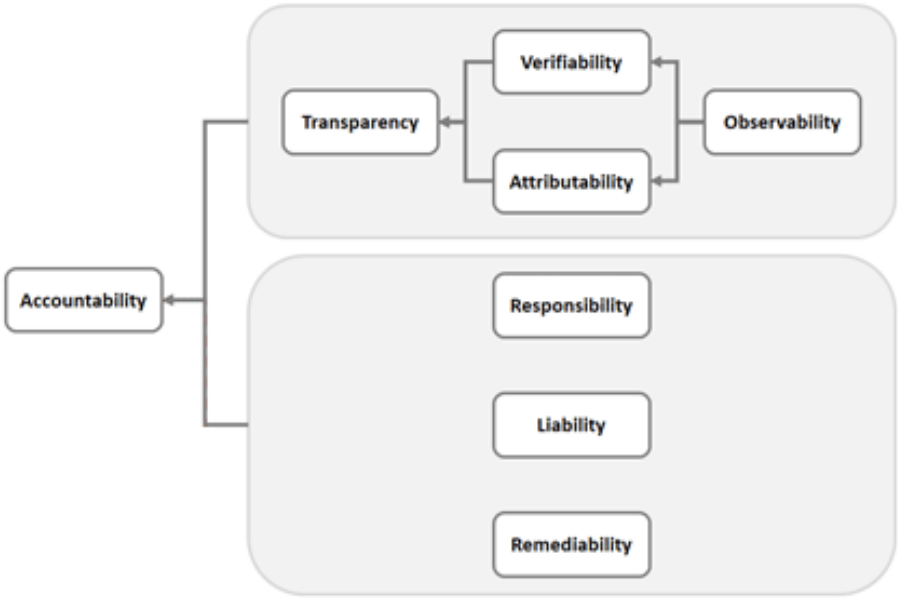


Figure 2: Accountability attributes[1]

Both verifiability and attributability rely on observability, as both ideally base on knowledge of internal actions of the system, but in fact rely on input and output to that system. The upper group is of primary interest in this work package, that is: observability, verifiability, attributability, and transparency, and it is reflected in the accountability evidence definition. The lower group, that is: responsibility, liability and remediability, are not basis for evidence, but rather rely on evidence.

2.1.2 Accountability practices

The A4Cloud Conceptual framework defines practices that are sets of behaviours that an organisation should have in order to be accountable, and are distinguished into four broad categories:
(i) Defining policies to comply in a responsible manner with internal and external criteria,
(ii) Ensuring the implementation of appropriate actions (including procedural mechanisms to ensure these policies get rolled out) to actualise such governance,
(iii) Explaining and justifying those actions, namely, demonstrating regulatory compliance,
(iv) Remedying any failure to act properly – means for remediation and external enforcement.

The core role of evidence is to support practice (iii) and also partially practices (ii) and (iv). Accountability obligations mentioned in WP: B-3 provide reflection of practices for use cases. The framework of evidence provides basis for analysing whether these obligations had been satisfied or not.

In the A4Cloud Conceptual Framework, three different levels of verification are suggested, which are organisational policies, IT controls and mechanisms, and operations. In the proposed framework of evidence we aim to approach verification through a standardisation of collected information and evidence format, from related events observed on any of those levels, linking the organisational and operational level of the cloud services with the mechanisms providing or supporting the services. It is important to notice that to provide automated verification of policies based on evidences at any of these levels, policies must have be defined in a machine-readable format. In A4Cloud the language considered for this purpose is A-PPL, developed within the WP C-4.

2.1.3 Cloud actors

In WP B-3 internal report MSB-3.1 (Bernsmed et al. 2014) definitions for the different cloud actors were analysed in terms of cloud computing roles; a summary is presented as follows:

[1] Based on figure of taxonomic analysis of accountability attributes, in “A4Cloud Project: MSC-2.3 Conceptual Framework“ (Felici and Pearson 2013)

- **Cloud customer**[2] is a person or organization that maintains a business relationship with, and uses service from, cloud providers.
- **Cloud provider** is a person, organization or entity responsible for making a service available to service consumers.
- **Cloud auditor** is a party that can conduct independent assessment of cloud services, information system operations, performance and security of the cloud implementation.
- **Cloud carrier** is an intermediary that provides connectivity and transport of cloud services from cloud providers to cloud customers.
- **Cloud broker** is an entity that manages the use, performance and delivery of cloud services, and negotiates relationships between cloud providers and cloud customers.

It is pointed also that an actor can have assigned more than one of the listed roles. However, only one role is attributed to an actor for each identified interaction among the actors.

An analysis of the cloud actors in relation to data processing and data protection roles lead to the definition of the following designations, that we will use extensively through this document:

- **Data Controller (DC),** is a natural or legal person, public authority, agency or other body which alone or jointly with others determines the purposes and means of the processing of personal data.
- **Data Processor (DP),** is a natural or legal person, public authority, agency or any other body, which processes personal data on behalf of the controller.
- **Data subject (DS),** is an identified or identifiable natural person (i.e. living individual). An identifiable person is one who can be identified, directly or indirectly, in particular by reference to an identification number or to one or more factors specific to his physical, physiological, mental, economic, cultural or social identity.
- **Data Protection Authority (DPA),** is an independent body which is in charge of: monitoring the processing of personal data within its jurisdiction (country, region or international organization); providing advice to the competent bodies with regard to legislative and administrative measures relating to the processing of personal data; hearing complaints lodged by citizens with regard to the protection of their data protection rights.

Identically, an actor can be assigned more than one data protection role, though not in relation to a particular set of personal data. For example, an actor can be a controller with respect to one set of personal data and at the same time be a processor with respect to another set of personal data, which it processes on behalf of another controller.

2.1.4 Obligations

The following list of obligations was collected from the same report, from WP B-3. We point to that report for a full list of those obligations that provides extended details, including legal perspectives.

List of obligations from the regulatory perspective (Data Protection Directive), to which Cloud actors must adhere:

- **Obligation 1: informing about processing**. Data subjects have the right to know that their personal data is being processed.
- **Obligation 2: informing about purpose.** Data subjects also have the right to know why their personal data is being processed.
- **Obligation 3: informing about recipients.** Data subjects have the right to know who will process their personal data.

[2] Another commonly used term with the same interpretation is "cloud consumer".

- **Obligation 4: informing about rights.** Data subjects have the right to know their rights in relation to the processing of their personal data.
- **Obligation 5: data collection purposes.** Personal data must be collected for specific, explicit and legitimate purposes and not further processed in a way incompatible with those purposes.
- **Obligation 6: the right to access, correct and delete personal data.** Data subjects have the right to access, correct and delete personal data that have been collected about them.
- **Obligation 7: data storage period.** Personal data must be kept in a form that permits identification of data subjects for no longer than is necessary for the purpose for which they were collected.
- **Obligation 8: security and privacy measures.** Controllers are responsible to the data subjects for the implementation of appropriate technical and organizational security measures.
- **Obligation 9: rules for data processing by provider.** Controllers are accountable to data subjects for how sub-providers process their personal data.
- **Obligation 10: rules for data processing by sub-providers.** The controller must also ensure that all sub-providers involved in the service delivery chain do not process the personal data, except on the controller's instructions (unless they are required to do so by law).
- **Obligation 11: provider safeguards.** Controllers are accountable to data subjects for choosing data processors that can provide sufficient safeguards concerning technical security and organizational measures.
- **Obligation 12: sub-provider safeguards.** The previous obligation comprises all processors in a service delivery chain.
- **Obligation 13: informed consent to processing.** Controllers are accountable to the data subjects for obtaining informed consent before collecting personal data.
- **Obligation 14: explicit consent to processing.** Controllers are accountable to the data subjects for obtaining explicit consent before collecting sensitive personal data.
- **Obligation 15: explicit consent to processing by joint controllers.** Controllers are accountable to the data subjects for obtaining explicit consent before allowing joint data controllers to process their sensitive personal data.
- **Obligation 16: informing DPAs.** Controllers are accountable to the data protection authorities to inform that they collect personal data.
- **Obligation 17: informing about the use of sub-processors.** Processors are accountable to the controllers for informing about the use of sub-providers to process personal data.
- **Obligation 18: security breach notification.** Controllers are accountable to data subjects for notifying them of security incidents that are related to their personal data.
- **Obligation 19: evidence of data processing.** Processors are accountable to the controllers for, upon request, providing evidence on their data processing practices**.**
- **Obligation 20: evidence of data deletion.** Processors are accountable to the controllers for, upon request, providing evidence on the correct and timely deletion of personal data.
- **Obligation 21: data location.** Data controllers are accountable to the data subjects for the location of the processing of their personal data.

2.1.5 Types of evidence

Activities in cloud services generate different types of evidence. Evidence types were investigated and identified in the previous task of this WP C-8. Five major types of interest for the A4Cloud project purposes are:

- **data processing practices**: evidence on operational practices such as replication, storage, deletion, copy, access, optimization, consent, security, segregation, proofs of retrievability, etc.;

- **data collection practices**: evidence on data collection practices such as policies compliance, privacy issues, security breaches, etc.
- **notification:** evidence that notifications were sent to the interested stakeholders in case of privacy issues (unauthorized access, etc.), policy violations, security breaches (data leakage, data lost, corrupted or tampered, etc.) and services or policy modifications, as well as service practices and users rights;
- **remediation:** evidence on remediation to their customers in case of security breaches, privacy issues and policy violations;
- **organisational practices:** evidence on requirements related to employee's training, system certifications, privacy policies, etc.

We list some examples of these types of evidence and relate them to obligations defined by WP B-3 whenever possible. Lists of referenced obligations and related actors can be found in above subsections.

Types	Sub-types	Sources	Obligations	Actors
Data processing practices	On data privacy	Logs, certificate, A-PPLE logs		DP
	On policy compliance	Logs, policies		DP, DC
	On data segregation	Logs (Storage Management System), contract, SLA	<Not defined yet>	DP
	On consent	Logs, metadata, policies	O1, O2	DP
	On data location	Logs (Storage Management System), geo-location, metadata	O7	DP
	On data replication	Metadata, logs (Storage Management System)	<Not defined yet>	DP
	On accessing of data	Logs, metadata, policies	O15, O10	DP
	On safeguarding data	Logs, metadata, cryptographic proofs, certificates	<Not defined yet>	DP
	On storage practices	Logs, metadata, geo-location	O5	DP
	On data deletion	Logs (Information Lifecycle Management), policies	O4	DP
Data collection practices	On policy compliance	Logs, policies	O13	DP, DC
	On collection practices	Logs, policies, metadata	O13	DP
	On privacy / security	Logs, crypto proofs, certificates	O13	DP
Notification	On data privacy	Logs, metadata	O6, O16	DC
	On policy violations	Audit report, logs, policies	O6, O16	DC
	On security breaches	Audit report, crypto proofs	O6, O16	DC

	On service / policy modifications	Logs, policies, contract, document, SLA, emails	<Not defined yet>	DC
Remediation	On policy violations	Audit report, logs, policies	<Not defined yet>	DP, DC
	On security breaches	Crypto proofs, logs, audit report	<Not defined yet>	DP, DC
	On privacy of data	Logs, audit report	<Not defined yet>	DP, DC
Accountability practices	Policies for inquiries and complaints	FAQ, compline, procedures	<Not defined yet>	DP, DC
	Maintain privacy policy	Privacy policies	<Not defined yet>	DP, DC
	Education and awareness	Document (about corporate education/training practices)	<Not defined yet>	DP, DC
	Data privacy breach and security risk	Document about organizational structure, privacy policies, privacy code	<Not defined yet>	DP, DC
	Third party risk	Contracts, agreements, policy, risk assessments, SLA	<Not defined yet>	DP, DC
	Maintain governance structure	Job description, code of conduct, report, ethics guidelines, risk assessments / reports	<Not defined yet>	DP, DC
	Maintain installation / operation process	Logs, documentation	<Not defined yet>	DP

Table 1: Evidence types.

2.2 Accountability Evidence

In this section we introduce the concept of evidence for accountability of cloud and future Internet services. We consider the context of accountable systems and subsequently we relate accountability evidence definition with key accountability attributes. We will investigate and assemble a definition and examine major concerns and requirements. These provide the basis for the framework of evidence.

2.2.1 Motivation for a definition

Evidence is one of the key elements connecting security aspects of digital information systems with the legal obligations such systems must meet. For digital systems the term evidence is most often associated with the area of digital forensics. However, the discussion of evidence definition in the context of digital systems is mostly missing from computer science research. Analysis of related work provides extensive knowledge of what can constitute evidence in specific practical problems. Studies also show that electronic evidence is more and more frequently permissible in court (Killalea and Brezinski 2002), (Volonino 2003). Related works use the term extensively, but do not attempt to discuss its definition.

In our opinion, the missing attempt to discuss a definition of evidence is a result of perceived sufficiency of intuitive understanding of the term, supported in some cases by general legal definition. This approach has proven successful enough in the area of digital forensics, mostly in our opinion, because the approach to evidence gathering and processing in digital forensics does not differ in general principles from traditional forensics. This approach can be characterized by looking for unintended evidence, i.e. evidence that some party was not planning to leave and which collection was not planned ahead (at

least for the purpose of forensics)(Park et al. 2012), (Kessler 2012). However, when this approach changes, the legal-based intuitive definition might not suffice.

Lack of mechanisms to verify, in real time, the security features implemented by cloud providers, and lack of auditability in real time or near real time are one of the major obstacles for large scale adoption of cloud computing ("Trend Report: Top Trends 2012-2013 - 70516" 2013; Park et al. 2012). Accountability for Cloud and other future Internet Services (Pearson et al. 2012) aims to address this challenge. Accountability can be defined in contrast to forensics as a planned process. Evidence collection is defined upfront and based on a carefully designed framework and metrics. This way the evidence is available at any time or stage of the service delivery process. This allows for a much wider spectrum of actions (e.g. prevention, compliance) and shorter reaction time. We argue that evidence process in the context of accountability would benefit from further discussion and formalization of the definition of evidence. Moreover, we think that such definition should be data centric, because data can be considered the basic universal unit of what comprises digital evidence.

2.2.2 Existing definition

At the time of writing of this deliverable, the A4Cloud glossary provides a definition of evidence included in the ENISA report: Privacy, Accountability and Trust – Challenges and Opportunities, (Castelluccia et al. 2011):

> "An accountability system produces evidence that can be used to convince a third party that a fault has or has not occurred (evidence)."

Originally proposed to characterize a property of accountability in that report, this definition is in A4Cloud project's context too minimalist and too reductionist in purpose. We propose the following definition of evidence in the context of accountability.

2.2.3 A4Cloud definition of accountability evidence.

> ***Accountability evidence*** can be defined as a collection of data, metadata, and routine information and formal operations performed on data and metadata, which provide attributable and verifiable account of the fulfilment of relevant obligations with respect to the service and that can be used to convince a third party of the veracious (or not) functioning of an observable system.

Data are defined as electronic records of processes in the Cloud or future Internet services. Metadata are defined as electronic records that describe how data have been stored and processed.

Routine information is defined as information that is provided by the organization about its internal routines e.g. implemented processes, employee training, etc. that can have influence on accountability evaluation.

Formal operations are defined as a collection of cryptographic and statistical algorithms, logic deduction, calculi and other methods that process data and metadata in a backtrackable manner. Backtrackable can be defined as deterministic or probabilistic but repeatable beyond reasonable doubt.

The definition is flexible enough to consider both a data-centric aspect, fundamental to an evidence collection, and information concerning active processes as organisational routines and formal operations, and respective outputs. Data, metadata, formal operations and routine information can be called the ***supporting evidence elements*** for accountability.

2.2.4 Privacy concerns with evidence collection

As stressed in deliverable D37.2, Privacy Design Guidelines for Accountability Tools (Onen and Pulls 2013), an accountable organisation should be a responsible steward for all personal and confidential data for which it is entrusted, including any new data generated as a consequence of it trying to be accountable. This means that the Framework of Evidence needs to be designed with privacy in mind. The EU Data Protection Directive 95/46/EC, Article 2, defines personal data as (emphasis added);

> (a) 'personal data' shall mean any information relating to an **identified or identifiable** `natural person ('data subject'); an identifiable person is one who can be identified, **directly or indirectly**, in particular by reference to an identification number or to one or more factors specific to his physical, physiological, mental, economic, cultural or social identity.

If data is derived from personal data to create evidence, then the data that constitutes evidence is most likely to also be personal data, since the evidence is related or relatable to an identified or identifiable person (the data subject of the personal data). There are generally accepted impossibility results of adequately anonymising data (*Differential Privacy*, Dwork 2006), thus assuming personal data is collected as part of evidence collection the only reasonable approach is to treat the evidence as personal data.

Recognising that evidence collection involves the collection of personal data the next step is to take into account that the evidence collection in and of itself may create new personal data and new risks. For example, aggregate insights into how evidence that relates to a data subject is processed may reveal everything from social-economic status to how frequent the data subject's data is audited, which may be an indication of the frequency of complaints issued by the data subject. Proliferation of personal data into evidence collection systems per definition means that personal data will be stored in additional systems compared to if no evidence collection system was in place. In addition, evidence may be retained for far longer durations due to legal or contractual obligations. Last but not least, the evidence systems will be most likely used by auditors, who would not otherwise normally have access to the personal data.

In light of these issues we stress that careful consideration needs to be given to evaluate the impact on privacy when evidence collection systems are deployed. Improper processes around evidence and excessive evidence collection risks undermining trust in a service. One does need not to look further than the recent developments around the EU Data Retention Directive 2006/24/EC to see the potential issues around excessive evidence collection. In A4Cloud, WP C-7 provides privacy design guidelines for accountability tools in deliverable D37.2, which may be used when designing evidence collection systems and to help identify potential issues when deploying such systems.

3 Requirements of Framework of Evidence

In this section we consider the generic requirements for the conceptualization of the framework of evidence, according to the definition given in Section 2, to support accountability for cloud services. We list the major concerns related to the evidence collection and application to automation of audit processes and verification of policies compliance and security breaches.

3.1 Major Concerns and Design Considerations

The concept of framework of evidence is founded on the following major concerns:

- To give account of events and occurrences within the services of cloud providers in a non-invasive manner to the customers' privacy and without exposure of sensitive material from the inside of organisations;
- To ensure information is collected securely in a tamper-evident process, which however may allow verification checks from different services, tools and trusted third parties (e.g. auditors);
- To avoid excessive data collection, minimising privacy issues and ensuring scalability of evidence storage with time, even with growing user bases and multi-tenancy scenarios.

And taking into consideration that evidence collected must be trustworthy enough to, if necessary, be admitted in a court of law.

Different aspects to be covered and supported by evidence suggest different methods for both selection of sources and collecting practices. With generality in mind, we will consider three main types of sources:

- internal data as logs, metadata SIEM outputs, etc., collected from monitored operations and services;
- data from end-users or from a cloud service to another, or internal data as customers' or employees' lists, training certificates, seals, etc.;

- documentation of the service's practices, contracts and organisational policies that relate to accountability and for which implied obligations can be expressed (manually or by other A4Cloud tools like AccLab or COAT) in machine-readable policies.

The information expected from each source differs in the purposes of evidence collection. Personal data raise mainly privacy concerns, critical in case of security breaches. Mechanisms and IT controls require support of correctness and demonstration of defined practices related with confidentiality, availability and security enforcement, and at the organisational level compliance with policies. The processing of evidence collection must have in account these differences in timing and purpose when selecting and gathering elements.

A critical property of accountability evidence is that attribution can be provided, i.e., the association between observed actions and the actors or components that execute them can be supported by evidence. As for other accountability attributes, the supporting evidence implies a more dynamic collection of elements directly connected with actions and service practices, than the case of forensics evidence, for example, where evidence is collected *a posteriori* of critical events. Accountability evidence needs not only to demonstrate unexpected behaviours and policies violations, but also to provide support for accountability and correct practices expected from cloud providers.

In this sense a mere gathering of information existing in a system may not fully ensure the complete collection of all necessary evidence, even considering a careful preliminary selection of elements at both operational and organisational levels. It is mandatory that the collected information can clearly correlate to validation purposes, and allow automation of analysis conformable in terms of performance to a continuous execution within the services.

To our understanding the most reliable way to realize this is a proactive approach, creating a direct response of the systems upon each event, after a continuous monitoring service, which records on the moment all necessary information (defined ahead by service administrators, for example). Such approach permits the services to access all elements that assure the connection between events, actors, purposes, times of occurrences and the applicable policies.

3.2 Privacy

The EU Data Protection Directive 95/46/EC, Article 2, as it was referred before, defines personal data. If data is derived from personal data to create evidence, then the data that constitutes evidence is most likely to also be personal data, since the evidence is *related or relatable* to an *identified or identifiable* person (the data subject of the personal data). For example, using a use-case from WP B-3 regarding the elderly patient Kim to illustrate it, the list of personnel that has access to Kim's health records is personal data of Kim. This is because whoever is treating Kim may reveal information about Kim, e.g., Kim may require special care for a knee injury by a specialist. Thus, evidence of who has access to (or has accessed) Kim's medical records is personal data of Kim, since such evidence would, e.g., reveal the fact that Kim has a health issue with his knees. Similar arguments as for personal data can be made for why data (evidence) related or relatable to business confidential data should also be considered business confidential data.

While recognizing that evidence can be both personal data and business confidential data, central questions in this work package are conceptually orthogonal to any privacy or business confidentiality concerns:

- What is the evidence that needs to be collected?
- How do you use the collected evidence to verify the correct behaviour for achieving accountability?

Determining what data needs to be collected and how to use it to verify correct behaviour does not necessarily constitute a breach of privacy or business confidentiality. Potential issues are encountered first when data is collected, how it is collected, how it is processed, who it is processed by, how is the data stored, how is the data shared, how long it is stored, to whom it is shared, etc. Therefore, from a privacy and business confidentiality perspective, the focus on the early stages should be on determining what to collect as evidence such that accountability can be verified (in part) with help of the evidence,

while recognizing from the start of our work that evidence does indeed constitute potentially both personal data and business confidential data.

Transparency of data processing is recognized as an important part of data protection, see e.g. Articles 7, 10, and 11 of the EU Data Protection Directive. Creating evidence of correct data processing and sharing it with data subjects is therefore in line with the principles of data protection. Sharing too much information about the processing may however risk revealing business secrets of the data processor, as also recognized in recital 41 of the EU Data Protection Directive. For achieving accountability, we note that one of the goals of collecting evidence is *to facilitate the detection of* violations. Conceptually, this means that the only information that needs to be provided to the verifying parties (parties towards whom an entity is accountable) is all privacy violations, not necessarily all generated evidence used to detect violations. Realizing this, the amount of personal and sensitive information about data subjects (or businesses) shared with verifying parties may be minimized, since only information related to violations would be shared. Presumably, it is in the interest of both data subjects and businesses to keep the accountable entity accountable, even if verifying parties (such as an auditor) learns (directly or through deduction) some information about them.

To summarize, giving data subjects detailed information related to their personal data (all derived evidence for example), is in line with the data protection directive in terms of making data processing transparent, however, it may be unacceptable for the entity performing the data processing. Giving any entity other than the data subject him or herself access to evidence that relates to the data subject may be a violation of the data subject's privacy rights. Conceptually, a party holding a cloud provider accountable only needs access to evidence that proves that a violation has occurred. Excess access to evidence beyond the bare minimum is a violation of the principle of data minimisation.

3.3 Integrity

The concept of data integrity, or in the particular case evidence data integrity, means that data has not changed, either by malicious actions or accidental causes (as hardware malfunction, e.g.) since its generation, gathering or generally after being classified as part of evidence. It implies that information is kept unaltered over stages of its entire life cycle, including any required transit and storage, for assurance of consistency, accuracy, and trustworthiness.

Integrity of evidence data also requires source integrity, meaning that source or origin of data must be clearly identified; that the link to the entity, person, organization or system component must be properly documented; and that its demonstrability is totally assured.

Without safeguards that can guarantee the integrity of data, the validity of data collected consequently becomes weak and cannot be used for the purposes of evidence, accountability support, and it may also become inadmissible in a court of law.

3.4 Verifiability

According to the A4Cloud conceptual framework, verifiability is the ability to prove that an actor behaves following a set of requirements originating from either contractual relationship or from legal obligations. Contracts and regulations may define the level of verifiability or even the type of evidence required during a regular verification process or even an auditing process. Evidence may be used to verify two types of behaviours: the Data Subject or the Auditor may verify an evidence in the case of misbehaviour, for example in the case of policy violations; whereas on the other hand, the Data Subject or the Auditor may periodically verify the correct behaviour of the Cloud Provider such as in the case where a cloud provider correctly logs actions taken during the processing of data.

Furthermore, verification of evidence can either be one-time request or periodic. The Cloud Provider may be requested to verify the correct storage of the data at a given time and to provide evidence from that specific time. In this case, an Auditor will only be able to verify the behaviour at that time without any proof of historic actions. On the other hand, verifiability can also apply to a long period of time. In this case, the Cloud Provider can be asked to provide evidence periodically during the required period; for example evidence on the history of actions.

3.5 Evidence Attributes and Requirements Summary

From the evidence collection and auditing point of view, the most important attributes of evidence can be defined as follows:

- Policy specific: Data collected for evaluation in audits is only useful as evidence, if it is tightly connected with the audit policy (Pretschner et al. 2009).
- Integrity: Evidence must be protected from manipulation of adversaries. This also includes internal adversaries (e.g., careless or malicious provider personnel).
- Origin: authentication of collector entity
- Chronological sequence: In order to produce audit trails, evidence and events captured in evidence need to be connected and back-traceable to a certain point in time (e.g., by using timestamps).
- Level of abstraction: Raw data can be hard to evaluate during audits, therefore a reasonable level of abstraction is needed for evidence to be useful in audits. This attribute is highly dependent on the type of compliance check.
- Dependability: The level of certainty in evidence is also important to consider. Casey proposed a scheme for certainty classification of evidence, which takes factors such as signs of tampering, number of sources, independence of sources and other uncertainties into account (Casey 2002).
- Automated verification oriented: Evidence analyser and Agents Audit System, AAS.
- Scalable: for storage and efficiency.

4 Framework of Evidence

Based on our definition of evidence and identified requirements for the framework of evidence, we aim to collect necessary elements from the services and underlying systems that can be used to support accountability in the context of A4Cloud project. This includes evidence relative to data stewardship (including processing, sharing, storing and deleting according to contractual obligations and legal requirements) with special consideration to data protection and confidentiality, correctness of services in terms of procedures and mechanisms, as well as support for notifications and for remediation upon detection of faulty behaviour.

We propose a model that deploys similar functionality and dynamics present in frameworks for digital evidence and audit process as: Accountability as a Service for the Cloud (Yao et al. 2010a), or Long-term trusted preservation service using service interaction protocol and evidence records (Blažič, Klobučar, and Jerman 2007). The framework presented can be extended by a secure log architecture, which is beyond the scope of this deliverable. Alternatively, the A4Cloud project provides tools, namely the Transparency Log (TL), which can be used to provide secure logging with extra privacy awareness.

Additionally, our model also considers cryptographic proofs and other demonstrative procedures resultant of formal, logical or mathematical algorithms, which may provide, periodically or upon requests, assurance on particular aspects of the behaviour of cloud actors. In particular, these proofs give assurance of the fulfilment of the obligations related to the processing and storage of data by CSPs. These obligations support, for example, the integrity or the correct processing of outsourced data. In conjunction with logs, cryptographic proofs intend to provide irrefutable evidence of the occurrence or non-occurrence of an action or an event [ISO/IEC 10181-4].

4.1 General Overview

The framework of evidence is based on the following assumptions:

- All actors or components acting in a service are identified, properly authenticated and the system contains demonstrative elements (e.g. logs) from the authentication mechanism(s);
- Events that change the state of the system (not only data related, but also at higher level e.g. business practices) are planned and expected as part of the life cycle of the services, reflecting contractual and regulatory obligations to be expressed in machine-readable policies;

- Cloud Providers act upon defined policies, and in a composition of services, policies are considered independent[3] in the chain of provision;
- A monitoring system exists that allows gathering of information implied by the above assumptions.

With these assumptions, the following general scheme is devised to collect and organise information for evidence provisioning.

i) Plan ahead the events and system sources to monitor, eventually based on expected evidence types and obligations expressed in machine-readable policies;
ii) For any action, operation or any change occurring in the system, collect descriptive elements of the event (what, when, whom, etc.) and elements from the system supporting the procedure (elements of evidence);
iii) Assemble the elements in an evidence record, including references for the supporting elements;
iv) Time-stamp the records by a trusted authority or trusted service, guaranteeing its origin and integrity;
v) Save the record within the cloud service provider, assumed securely by any safeguards the provider considers for protection of its own data.

The event's records shall constitute accountability evidence, and may be made available to related stakeholders (as data subjects in what concerns to their data processing), auditors and regulators.

Two distinct steps are foreseen. First, monitor simultaneously the listed types of sources and expected events in the different timings within the services' lifecycle, recording the state of those sources with an event description for any change observed. Second, analyse the information collected to detect, as early as possible, policy violations and breaches, and store the acquired information.

The output of this framework will be a set of records to be stored on the side of the provider only, describing events and changes in the system, with a format planned carefully to satisfy the requirements of evidence. The elements in the system that support the actions observed will not be stored with the record, but merely referred by the location/name and a cryptographic digest, avoiding big amounts of duplicate data. Exception may be the cases were these supporting elements might be of special importance, as is the case for policy violations or security breaches, and a copy may be attached to the record, eventually encrypted for security. An automated auditor system takes care of the continuous processing of elements and collection of records, based on a constant monitoring of the providers' systems and interactions with policy enforcement tools.

The description of an event shall contain enough elements to compare with defined policies and agreed obligations, complemented by references to existing sources in the system that document the changes or actions observed (as logs, cryptographic proofs, output of specialized toolkits, and other documentations describing data processing and service activities and practices).

Chains of records, time and action bounded, associated to a same resource (typically data from service users or internal data from the organisational level of the cloud provider) are analysed on a continuous base by an automated system, possibly with support of a knowledge-base and an inference engine oriented to policy verification. Non-compliance of recorded events with applicable policies, security breaches or any detection of unexpected behaviour must make the automated auditor system trigger alerts to the services, incident management tools, and finally report to interested stakeholders.

External auditors may be granted access for verification of service practices through the evidence repository, and the framework of evidence provides procedures for validation of the records' data integrity, origin and time-creation.

[3] Even in case of different providers acting upon the same data they most probably will have different purposes (e.g. data collection, data processing, data storage) reflected in different types of policies.

An overview of the framework of evidence is presented in Figure 3, and the details of this framework and its dynamics will be developed in the following sub-sections.

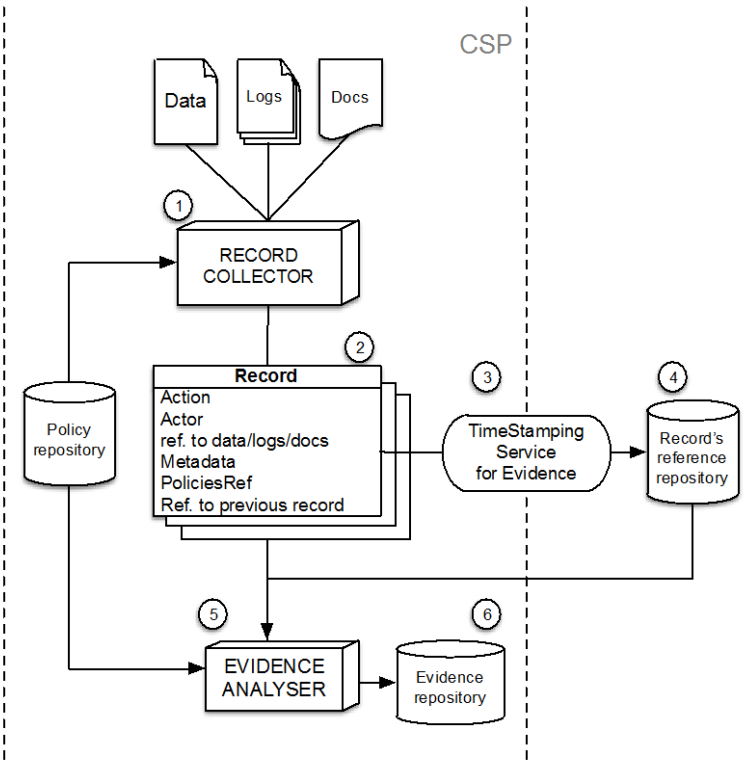


Figure 3: Architecture of the Framework of Evidence

1. Record Collector – It gathers descriptive elements related to changes occurring in the service, and references to supporting elements as logs, data and services' documentation, collect them in a pre-defined format, which we formally denominate as record.
   A policy repository, external to the framework of evidence, is assumed to be accessible, allowing applicable policies to be referenced.
2. Record – Compiled in a continuous manner to give account of service procedures, either triggered by actions or periodical checks. Its format tries to encompass a minimal description of an event or a state of the system at a given moment in time.
3. Time-Stamping Service for Evidence. – Service that creates time-stamped references for the records, digitally signed by the provider collecting evidence, consisting of a small hash digest and linking event-related records referent to a resource (usually data and associated subject).
4. Record's reference repository – Append-only storage for the references generated in the previous step. These values will be used as reference in the records' linkage accessible from outside the service to verification. This approach is intended to assure integrity of records and fixate supporting elements of evidence as logs, for example in auditing processes, upon recalculation of the hash digests and comparison with the external values.
5. Evidence Analyser – Takes as input chains of records, event-linked, and applicable policies. It validates the actions described against the obligations and rules expressed in the policies, identifying possible violations and irregular procedures, or otherwise supporting the adequacy and correctness of the event.
6. Evidence Repository – Evidence is stored (by cloud provider) as associated chain of records. After analysis and validation, it may support the good functioning of a service or demonstrate detected faulty behaviours, either in internal routine checks or by external auditing process.

Each action or operation is registered in a record, compiled with the elements collected from different sources. Such elements and sources must be defined upfront in order to ensure a complete description of the events and identification of entities involved, and will eventually depend on the type of data and type of evidence anticipated. Records act as a wrapper of those existing elements of evidence, in order to simplify the systematization of information gathering, and at same time form a simple component for later access and verification by auditors, regulators or third parties.

The records system is based on similar concepts of digital evidence formats such as Evidence Record Syntax - ERS (Brandner, Pordesch, and Gondrom 2014), Digital Evidence Bags - DEB (Schatz and Clark 2006; Turner 2005), among others, adapted for accountability purposes and specifically to a later verification process of policies in A-PPL format. An event can be considered as a succession of actions or service operations that are well described by a time-linked chain of records.

4.1.1 Cryptographic proofs and other on-the-fly evidence

As far as cryptographic proofs are concerned, they may be requested by the data subjects or external auditors as specified in a policy. Data controllers and/or processors generate these proofs on receiving such requests and send them back to the requesters for verification. Cloud actors can also keep a copy of the proofs as supporting evidence in a record for the occurrence of an event. Verification of the proofs is usually a task performed by the data owner, the data subject or an external auditor. The result of the verification process can also be recorded in our framework of evidence.

Cryptographic proofs provide an instantaneous assurance of correct behaviour from CSPs, verifiable at any point in time. Legal and contractual obligations makes such proofs of paramount importance. Indeed service providers are responsible to providing appropriate security and privacy safeguards to the data they collect, process or store. Similarly, data controllers are also responsible to provide upon request evidence on their data processing practices.

Cryptographic proofs should convince any entity that verifies them about the correct processing of data and should not be disputable. Therefore, they fit well in the record framework described above. Indeed, during the generation of these proofs, corresponding records can be generated and stored according to the record framework so that during an audit that involves the verification process, one can be ensured that the proofs cannot be repudiated.

Cryptographic proofs are of several types and address particular issues in cloud computing. To the best of our knowledge, there exist two kinds of proofs that focus on two services offered by the cloud delegation of storage and delegation of computation:

1. **Cryptographic proofs of storage** – Data outsourcing is one of the basic services supplied by the cloud. One may be concerned by the integrity and the availability of its outsourced data stored in the cloud. Cryptographic proofs that aim at remotely ensuring the correct storage of data may consist of an evidence of the compliance with obligations related to data integrity and availability. A basic technique involves Message Authentication Codes (MAC). These cryptographic checksums represent the evidence that the data has not been tempered with. The data owner computes the MACs on its data (using cryptographic keyed hash functions) and stores both the data and its MAC to the cloud. At a later time, a verifier, who wants to check that the outsourced data has not been maliciously modified, retrieves the data and checks the data with the retrieved MAC, thereby providing the integrity assurance. However this solution has an important drawback in the context of cloud computing: high costs. The verifier has to retrieve the whole data to perform the integrity check, inducing important communication costs. One may think of dividing the data into blocks and to compute the MACs on each block. But this modification induces an important storage burden at the cloud since it has to store a number of MACs linear to the number of blocks in the data.

   Recent techniques address the efficiency issue faced by the MAC-based solution. These techniques generate the proofs of storage based on a request sent by a verifier, and ensure with a high probability that the cloud stores the outsourced data as expected by contracts and

regulations. The notion of Proof of Data Possession (PDP) was introduced by Ateniese et al. (2007) and enables a verifier to verify the integrity of its data outsourced to the cloud in an efficient way, that is far more efficient than the solution presented above. In a PDP scheme, the data owner computes homomorphic authenticators as check-values for each data block. To verify that the cloud still possesses the outsourced data, the verifier asks the cloud for tags of randomly chosen blocks. The cloud generates a proof based on the targeted blocks and their respective authenticators. Due to their homomorphic properties, the cloud is able to aggregate the proofs leading to optimized communication costs.

Similarly, the notion of Proof of Retrievability (POR) was introduced by Juels and Kaliski,Jr. (2007) and adds to PDP an additional property which is retrievability. The data owner can actually recover the outsourced data even if small errors are detected in the data. This is possible thanks to the application of an error-correcting code to the data before its outsourcing. The PDP and POR solutions have the advantage to optimize the costs on both sides: the data owner only needs to store the keying material that was generated before the outsourcing and the data and the cloud performs light computations to build the proofs. Besides, the request for proofs and the responses to that request induce light communication overhead.

2. **Cryptographic proofs of computation** – The other main service offered by cloud is to provide computing resources to “weak” customers whose resources are limited. These clients are able to outsource their computationally intensive tasks to the cloud. However, they lose their control on the outsourced computation and they lack transparency from the cloud. Therefore they should be enabled to verify the correctness of the result of the outsourced computation with the requirement that the verification costs do not exceed the ones induced by the outsourced task itself. A certain number of techniques provide evidence for correctness of the result. This aims at proving the compliance or the non-compliance of the cloud provider with the obligation to process the data as specified in policies.

In this work we are particularly interested by Proofs of Retrievability. We think they may consist of valuable evidence for the compliance or non-compliance with cloud’s obligations to the correctly store the data outsourced by their customers. We focus on and integrate the PORs in our framework of evidence. Nonetheless, we think that cryptographic proofs of computation should also be provided as possible evidence so we keep them as future work.

4.1.2 Auditing

Common cloud computing scenarios usually include at least one cloud service provider and a customer in the service provision chain. However, how, when, by whom and where data are processed is usually unclear to the customer. Audits can be used to “verify conformance to standards through a review of objective evidence”. In cloud computing this means cloud services are independently examined regarding their performance, operation, data privacy, and security controls, which shall assure the cloud customer that appropriate measures are in place. The introduction of a third party auditor (TPA) offloads this time-consuming and cost-intensive task, as well as required knowledge to conduct audits from customers.

Auditors

Auditors can be internal or external. External auditors are members of independent organizations specialized on performing audits. Additionally, customers of cloud services can also take on the role of an auditor if they use public audit interfaces provided by the cloud provider. Internal auditors are usually contracted to the organization, whose entities are to be audited. They generally adhere to the same standards as external auditors with respect to performing an independent analysis.

4.1.3 Evidence in service delivery chains

Cloud service provision can present complex scenarios, where multiple service providers are chained for composition of services. There may be horizontal chains of SaaS providers, as well as vertical chains down to PaaS and IaaS providers. Additionally, some services might span multiple roles. So, within the service provision chains, Cloud Service Providers (CSP) may have multiple roles, i.e. they can be both cloud customers and cloud providers. Typically, these chains are hidden from the customer. The customer has no influence on the choice of third party services chosen by the CSP. Also, details about the actual integration of such services are not made transparent beyond documentation of parties involved in service provision. Cloud customers want to ensure that service delivery chains of cloud providers are in compliance with policies and SLAs.

A scenario analysis of service delivery chains was proposed and investigated in the previous internal milestone, MS C8.1, involving decomposition in different views, technical, obligation and evidence. It considers major aspects of delivery chains such as data location, retrievability, policy compliance and notification of breaches along the chain. As in most common real-world situations, it was considered that the cloud customer only has direct communication with the Primary Service Provider (PSP) and does not have direct communication with the CSPs, which in turn may not recognize that cloud customer as their client. In this cloud ecosystem, a cloud auditor was introduced to monitor the activities between cloud customer-PSP and PSP-CSP.

From a conceptual point of view of evidence collection, analysis and presentation for the interested stakeholders, this kind of scenario has its major issues relatively to privacy of data, business confidentiality and share of information to third parties. Assuming the auditor is a trusted entity agreed by all stakeholders as a trustable mediator, it is unconvincing to consider that organisations will be willing to lightly share all information about their internal business practices and strategies (even with the ultimate gain/stimulus of providing accountability of services and practices).

In that sense the framework considers the gathering of evidence on a service provider basis, in the format that we will introduce, collected to audit and presented (or made assessable) to the interested stakeholders in a reduced form, which minimizes the amount of information exposed and to whom is exposed.

In a chain of service suppliers the relation between any two actors is well defined. Each one will be either a consumer or a provider of a given service. An audit trail in a Service Delivery Chain must in that case be comprised of the composition of the audit trails from each cloud provider. A time and event bounded chain of records, providing chronological sequence, constitute the audit trail from a single provider, with the complement of evidences collected on-the-fly (e.g. proofs of retrievability, so important in the context of scenarios where data can be stored deep in the supply chain). The records' chain reflects the actions and obligations of that single provider, and the records provide the associated timestamps. From the perspective of multiple providers, the full sequence of events, can be properly reconstructed from the different chains aligned and compared, reflecting the recorded timestamps, and verifiable (including the origin and integrity of source) by the external hash chain or a Time-Stamping Authority.

We suggest the presentation of information describing process or operations to authorised and contractually specified actors only (as Auditors or Data Protection Authorities) with the assurance of existent supporting elements within the organisations that can demonstrate the occurrence or not of the recorded actions, but only to be presented in special cases of contestation or litigation. Evidence referring to data processing, may be presented exclusively to their owners and, as it was presented before, accessed by designated resource (a specific single element or a subset of data). It important to stress that this information, originated eventually in different providers and presented to Cloud

Costumers through the Cloud Auditor must necessarily be defined, stated in SLAs and with agreement of all involved part upon legal contractual basis.

4.1.4 Evidence integrity and alternative approaches

Records, being produced internally by the service providers, will need to demonstrate the origins, time of compilation and integrity to auditors and authorized stakeholders. For that purpose we propose a process of records' public reference based on time-stamping linked hashed-chains, similar to schemes proposed by (Buldas et al. 1998; Buldas, Lipmaa, and Schoenmakers 2000). The external publication of the hashed references (accessible for authorized verifications) is suggested as a way to assure verifiability of data integrity in the records, avoiding the trust requirement in a third part such as a time-stamping authority (TSA), or even its need. Later we will detail the time-stamping references' process and the linkage of records. Evidently, a traditional TSA can always be considered for the same purpose, involving an extra entity in the cloud ecosystem.

The model proposed can be implemented on top of a secure log architecture that allows verification and auditing, usually contemplating is own methods for integrity of logs entries (a record can be seen as a log entry in services' top-level process logging). As examples we point to architectures exposed in Logcrypt: Forward Security and Public Verification for Secure Audit Logs (Holt 2006), Efficient Data Structures for Tamper-Evident Logging (Crosby and Wallach 2009) or A Secure Log Architecture to Support Remote Auditing (Accorsi 2012). Those frameworks are oriented to at same time offer a high level of security and permit audit verifications without issues with data encryption and sharing of private keys.

Alternatively, A4Cloud tools like Transparency Log (TL) can be used for providing secure logging with a privacy-aware evidence collection in mind, and of special interest for multiple cloud providers in a supply chain.

4.2 Records Collector

The framework proposed aims to capture the dynamics of the services, changes in data and internal logs, with the creation of digital records that contain or refer to those elements. Records will act as a wrapper of the existing elements of evidence in the system, in order to simplify the systematization of information collection, minimize exposure of sensitive data, and at the same time form a simple component in later accesses for verification by auditors, regulators or third parties.

The approach suggested here is the definition of records based on usual techniques in digital evidence collection, incorporating a system of predefining the information to monitor (eventually to be implementable via an API to system administrators) and the frequency of the monitoring process. Without the need of rigidly establish what to be logged ahead, that choice can be made by the service architect or administrator, allowing to specify which logs, keywords and tokens to be monitored and in which frequency, according to the needs of the service or the accountability practices to be supported by evidence.

Logs, internal documents, or any private data, constitute the supporting elements of occurrence (or not) of actions and operations and its validity (or not), and we name them in this context as the **supporting elements** of evidence. Due to its sensitivity while private data, we propose to not include them explicitly in the records, but only its hash digests, calculated by a cryptographic hash function[4]. These small fixed-size bits of data are saved and used to refer the original elements. The use of hash functions applied to any element of evidence existent in the system produces small size references, avoiding expose original data, and constituting a way to fixate those elements in time and providing an easy verification method (by direct inspection of its digest against the recorded version), even if the name or location changes.

[4] Cryptographic hash functions (NIST - Information Technology Laboratory 2012), are one-way functions that generate from a given arbitrary piece of data a fixed-length value (also commonly designated checksum, digital fingerprint, hash digest or simply hash). Cryptographic hash functions are ideally collision resistant, meaning that two different sets of data should produce different values.

Moreover, any modification will be unavoidably detectable; a hash recalculation will inevitably result in a different value.

These digests are saved to a record with any other relevant information (detailed in next subsection). The full record with a timestamp will, in turn, be in its totality hashed and the hash value digitally signed and stored as a linked chain in an external publically accessible[5] repository. Similar approaches are proposed in the context of accountable network storages (Yumerefendi and Chase 2007) and time-stamping schemes (Buldas et al. 1998), the former including some suggestions for storage implementations (out of the scope of this work). Assuming that this approach ensures authenticity and tamper-evident records (no alteration can be done without immediate detection by hashes dissimilarity with the ones on the external storage), the integrity of the contained digests of private data and logs is also assured, and as such providing authenticity for that data without exposing it. In an audit process or cases of litigation, only the exhibition of the originals will produce the same hash stamps as the ones available publically in the records.

4.2.1 Records format

A record of an action can be seen as the atomic constituent of evidence. Enough elements must be gathered in order to characterize the state of the system and undergoing operations, which will be organized in the pre-defined form for standardization purposes. Records will need to be machine-readable (and compatible across systems supported by this framework) for automated verification and auditing processes. Actions will need to be matched against applicable policies and obligations expressed in those, for compliance validation, suggesting that collected elements and policies format should much as possible to simplify the automation of the validation process.

The elements to collect for each attribute of a record must be enough to describe the action or operation in a way that allows at any time, from the evidence collection, a reconstruction of each step of an event or succession of events. Such reconstruction is feasible if a link is accessible between every state of the system, or in this framework of every record. Records must be accurate, unambiguous and trustable, its integrity and authenticity must be irrefutable within the boundaries of reasonable, to demonstrate the course of events in a backtrackable manner. We based on models with similar proposals like the ones proposed in (Yao et al. 2010a) – timestamped traces, (Blažič, Klobučar, and Jerman 2007) ERS with a certification service, or (Turner 2005) – tags of digital evidence bags expanded by ontologies for correlation with events (Schatz and Clark 2006), and adapt the concepts to attain such type of records in the context of the A4Cloud project.

An event can be well represented by a characterization of the following attributes: the action description, the identification of who's performing the action, and the new state of the data or logs appending after the action, metadata and, for efficient purposes in later verifications, the applicable policies.

We suggest the following composition of attributes:

- $\mathbf{Action} = (actionID, timestamp)$
- $\mathbf{Actor} = (agentID(, purpose))$
- $\mathbf{SupportingElements} = \{hash(element_1), hash(element_2), \dots\}$
- $\mathbf{Metadata} = \{ meta\_element_1, meta\_element_2 \ , \dots \}$
- $\mathbf{PoliciesRef} = \{ PolicyRef_1, PolicyRef_2, \dots \}$
- $\mathbf{linked\_reference} = hash\_Ref(R_{i-1})$

In detail:

Action. For the description of actions or any operation performed, a categorization may be needed (read/write/delete/send, e.g.) extended with the timestamp to ordering and reconstruction of steps.

[5] By publically accessible we mean readable by actors related to the service, interested stakeholders and auditors or regulator entities.

$$\text{Action} = (actionID, timestamp)$$

Actor. The identification of the authenticated agent (subject, employee, operator or application component if an automated process), possibly extended with the purpose (as described in A-PPL specifications), if available.

$$\text{Actor} = (agentID\ (, \square\square\square\square\square\square\square))$$

SupportingElements. With integrity and storage scalability in mind, we propose to just reference any supporting elements of evidence present in the system (logs, organisational documentation, etc.) by its hash digests. The digests ensure that any originals are referenced in a tampered-evident manner (any change in the originals will be detectable by the different digest), avoiding direct exposition of private or sensitive data. Eventually complementary information to append to the hash fingerprint is the location of each original element. The set with the supporting elements is then:

$$SupportingElements = \{element_1, element_2, \dots\}$$

With

$$element_i = \left(elementLocation_i, hash(elementSource_i)\right)$$

Metadata. Information about data and system properties can be gathered directly from the system. This attribute provides extra details for each state and possible support for data processing and policy compliance validation. It is expected to be extracted from the system or specific logs, specified upfront the record collection and according to the service. A simple parsing application or a small data-mining task, is suggested to perform the extraction.

$$\text{Metadata} = \{metaElement_1, metaElement_2, \dots\}$$

A minimal set of metadata must contain sufficient elements required for services verification and compliance validation. For example, metadata of major importance are:

- timestamps of creation/input of data (how long is stored),
- last accessed (was read),
- last modified (was processed),
- Host name and location of data,
- geo-location of storage (policy compliance),
- others (to be defined according to the service and evidence type)

PoliciesRef. References to applicable policies, in principle retrievable from services' policy repository, will provide access to the policies' elements, allowing the matching of action and information from metadata against the obligations, access restrictions or any rules expressed by the policies.

$$\text{PoliciesRef} = \{PolicyRef_1, PolicyRef_2, \dots\}$$

This information gathered is to be processed into a standardized set of data, composing a record. Standardization ensures that independently of the time when a record is assembled, if the data hasn't been modified or accessed, the hash values calculated will be constant and cannot be reproduced or repeated without the same record elements.

Linked_reference. Any record describing an action upon a resource, links to the previous actions through the hash reference of the previous record (already calculated and stored).

$$\text{Linked reference} = hashRefTo(R_{i-1})$$

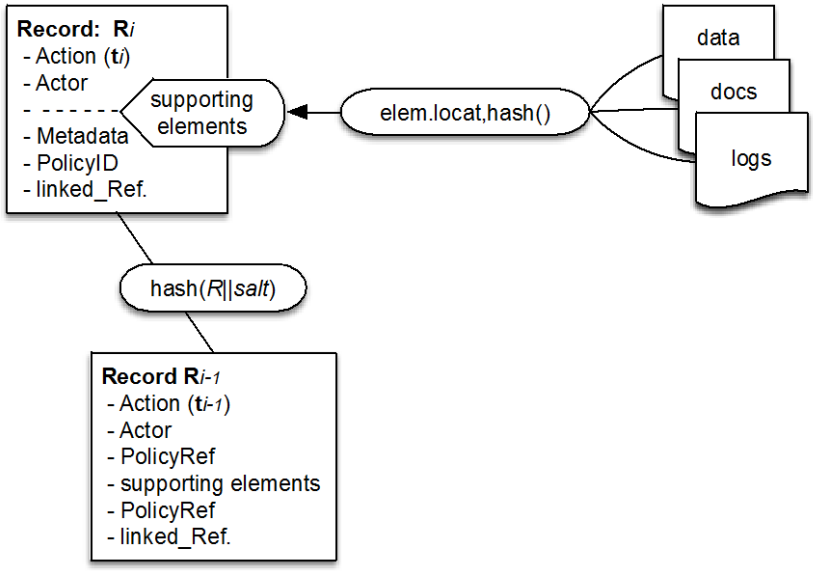


Figure 4: Records compilation and linkage

The hash references of each record are digitally signed and published to an external repository[6], with open access to interested stakeholders and auditors, guaranteeing that the records have an identified origin and can not be tampered without detection. Identifying origins provides information about the origin of data across the service chain, along with the identification of the actors operating in the different services within the chain of provision. The reference to previous records in the sequence of changes will assure time and event linking of records, which will be the topic of the next section.

4.2.2 Chaining of records

In this section we will provide further details on the linkage of the records in chains.

The linked reference is made with aggregation of a timestamp t_i, the identification of the entity compiling the record ID_i, with the calculation of the hash of the record itself. A secret salt[7] (known only by the service provider) may also be included in the hashing process; in order to compromise brute-force attacks attempting to determine the records' contents. The linking process is complete by including the previous reference in the chain.

A record R_i will have linked reference H_i:

$$linked\ ref(R_i) \equiv H_i = (t_i, ID_i, hash(R_i^*), H_{i-1}) \rightarrow [public]$$

And

$$R_i^* = R_i || Salt_{secret}$$

The contained reference to the previous record, H_{i-1}, provides the link to other records associated to the same resource and subject, eventually processed by different services for different purposes within the delivery chain. These references are immediately stored, after calculation, separately from the internal records' chain, in the external record's reference repository, and digitally signed by the service provider.

[6] Cf. previous footnote.

[7] Salt, in cryptography, is a set of random data, ideally long, used as additional input to a hash function, usually applied to hashing passwords, primarily to defend against attacks that attempt to determine the argument hashed by systematic computation of possible values.

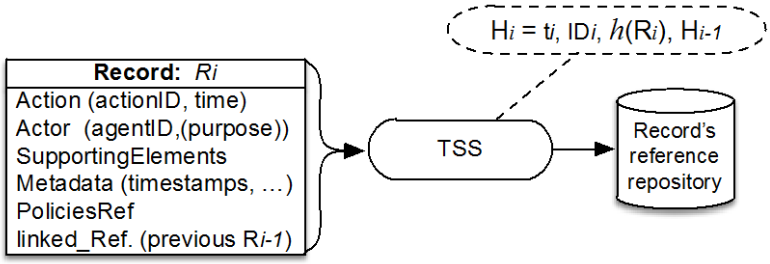


Figure 5: Publishing a timestamp reference of a record

The repository shall also contain the needed information for the relations between the signed digests, in this way bonded in chains, i.e., the reference to the parent record and respective branch.

Consider some data within a system (usually end-users' data), as any other elements as system logs, organisational docs, etc., pre-defined as potential source of evidence to be monitored. The original state of the data is considered the state zero or the root of the record tree to be associated with this particular data.

For any action (or at a specified lapse of time) a new record will be generated. As it was described before, records use as references tuples that ensures a link to related previous records. The first record may use (since there will be no previous reference) the hash digest of the subject ID and the data related to the action (that we will refer as subject's resource), defining a key-index for the origin of that particular chain.

The initial state, $S_0 = (subject||resource)$, will produce a record R_0 with a reference:

$$H_0 = (t_0, ID_0, hash(R_0{}^*), hash(S_0{}^*))$$

A second record will map an occurring change (or a time lapse in a periodic monitoring) and its hash reference will contain the hash of the previous reference, providing linkage and securing it of any attempts of tampering:

$$H_1 = (t_1, ID_1, hash(R_1{}^*), H_0)$$

And successively, for a generic state of the system, a publically assessable linked reference as it was presented before:

$$H_i = (t_i, ID_i, hash(R_i{}^*), H_{i-1})$$

The dynamics of such system, i.e. its progression in time, is represented in Figure 6.

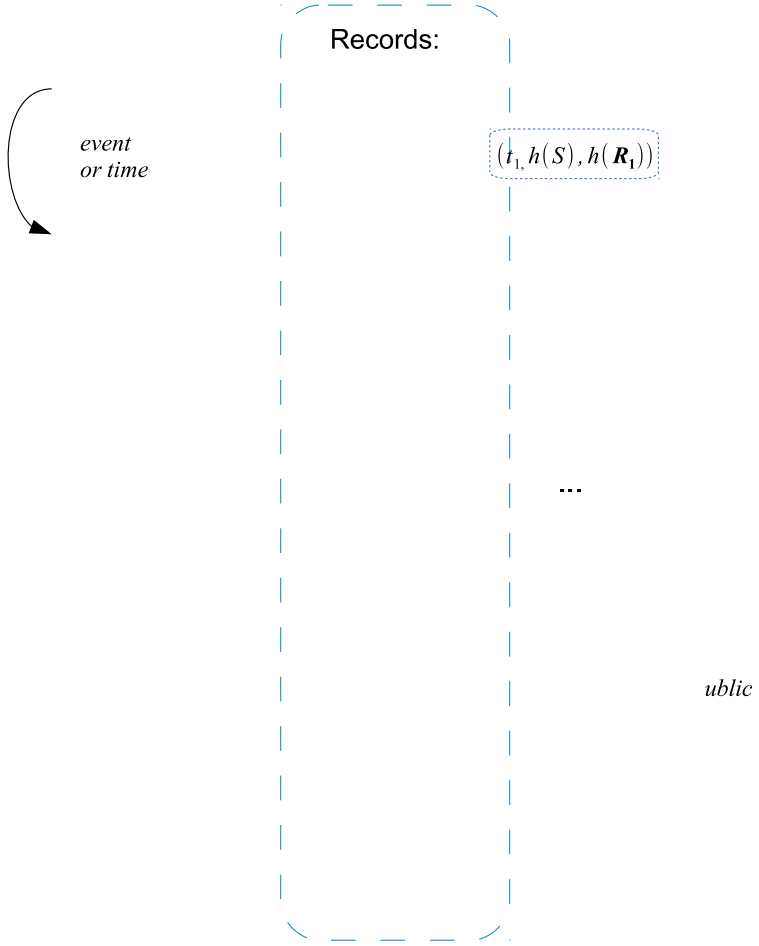


Figure 6: Authenticated chain of records

The process iterated with time will produce a succession of records (stored internally) linked in a chain by the (exclusive) hashed reference of the preceding records. This process is linear with the number of records, both for the reference creation as the verification process, and presented here in this form for the sake of simplicity. Optimizations have been proposed in the field of linked timestamping (Buldas, Lipmaa, and Schoenmakers 2000; Blibech and Gabillon 2006) that include binary linking schemes, connected graphs, skip lists (with logarithmic time).

The following abstract example shows more formally how data can be protected from tampering and exposure. Consider some private data, *D*, to be processed according some stipulated agreement (with conditions specified in a policy) resulting in some news state of data, D. The process, *P*, will be logged by the system, and the log *L* is considered itself sensitive data by the organisation.

$$P(D) \rightarrow D'$$
$$\log(P) \rightarrow L$$

With the process, a record and a hash reference are created:

$$actionID = descriptor(P)$$
$$R = \{actionID, \dots, h(D'), h(L), \dots\}$$
$$H_R = h(R)$$

The system stores the record internally, and the hash-reference H_R is automatically published, with a timestamp and digitally signed, to the external reference repository. After that procedure, any change in the record will be unavoidably detectable; a hash recalculation will inevitably result in a different value and as so revealing a discrepancy with the external H_R.

Likewise, mischievous changes in original data or logs will necessarily lead to different hash values, mismatching the respective information in the record, which by the previous statement can not be tampered to match the hash output from an improper changes.

The result of the process is an authenticated chain of records, with each record containing a traceable hashed digest of data, metadata and other evidence elements, plus a hash reference to the previous record.

The concrete benefits and immediate advantages of this framework for evidence are:

- Third party audits can be done upon the information collected in the records without immediate access to evidence sources (elements' integrity is assured by the hash references; and in ultimate cases, like litigation or forensics analysis, logs or other elements can be provided with restricted access)
- The standardisation of collected elements of evidence records eases validation of operations against the access rules and obligations expressed in the policies, an in special in scenarios with multiple actors running their own framework implementations. Verification can be done either internally (for routine checking, e.g.) or externally (for auditing processes or data subjects inspection), with a clear form of information's synchronisation and cross-reference.
- The amount of space necessary for evidence storage is limited by the use of hashing and chaining. It is important to note that it is not necessary for auditors to store related logs and data (Cloud provider is responsible that his logs are consistent with provided records).
- The publishing of the signed digested references ensures that data has its origins defined and cannot be tampered unnoticed, fixating logs or any other element of evidence (non-repudiation).
- Events can be back-traceable in time following the chain of evidence records linked by the hash references, allowing auditors and Data Subjects to verify performed operations or simply follow the location of data, using the published hash references' chain (attributability).

From an auditing perspective, a chain of linked records form an audit trail that can be analysed at any moment, in an automated way. The integrity of records is verifiable by comparison against the external linked references, and both internal or external audit processes may access (or request access) for verifications of policy compliance or services' correctness; all states of the system are back traceable, and matching the actions recorded against the obligations expressed in the applicable policies can identify violations and security breaches. External auditing can rely solely on information provided in the records for compliance verification and service correctness, eventually accessed by specified chains according to resource or subject (confidentiality), and the more sensitive elements of evidence are only requested if violations or breaches are detected, or if a closer inspection is necessary and agreed by interested stakeholders.

4.3 Proof of Retrievability (POR)

A cryptographic proof of retrievability was developed as part of the A4Cloud project as a demonstrative evidence of retrievability of data for data subjects and cloud services' end-users. That is a case of a formal operation, a cryptographic algorithm intended to be used straightforwardly by the data owners. This procedure is a dynamic operation, requested to the cloud provider as many times as the data subjects desires, and resulting in an immediate and direct response – the retrievability proof.

Legal and contractual obligations justify the need for POR. Indeed, controllers are responsible to the data subjects for the security and privacy of the personal data they collected. POR provides a way to check the integrity of the collected data. Moreover, processors are accountable to the controllers for, upon request, providing evidence on their data processing practices. We thus consider POR as evidence of data processor storage practices.

In a nutshell, a POR protocol uses cryptographic techniques that allow a verifier to check that their data is correctly stored (meaning that it is not modified nor deleted) without having a copy of the file in its

local storage. POR is a probabilistic solution in the sense that the assurance guarantee depends on the probability that the data is tampered with.

A POR protocol consists of an interaction between a verifier (data subject, data owner, auditor) and a prover (the entity that stores the data):

- Before outsourcing the data, the data owner processes the data to enable the retrievability verification (e.g computation of metadata such as signatures, check values, sentinel values, that prove the retrievability)
- At certain point of time, the verifier sends an evidence request for retrievability of that particular data to the prover.
- The prover responds with the proofs that are targeted by the request.
- The verifier analyses the received proofs and makes a decision about the retrievability of the targeted data.

The format of these proofs is generally digital signatures. They can also come with blocks of data that are used for computing the integrity checks.

StealthGuard is proposed as an efficient and provably secure POR scheme. StealthGuard makes use of a privacy-preserving word search algorithm to search, as part of a POR query, for randomly-valued blocks called watchdogs that are inserted in the file before outsourcing.
Thanks to the privacy-preserving features of the word search, neither the cloud provider nor a third party intruder can guess which watchdog is queried in each POR query. Similarly, the responses to POR queries are also obfuscated. Hence to answer correctly to every new set of POR queries, the cloud provider has to retain the file in its entirety.

4.4 Evidence Analysis and Verification

After a specifically planned validation process, a record or a chain of records can be considered evidence, as it was defined before. This analysis and verification process will be achieved by an analyser procedure, which will validate the collected information against the applicable policies, verifying compliance, fulfilment of obligations, assurance of data privacy, and ultimately correctness of service provision. An automated auditing system can execute this verification process in a continuous base or, according to the services and SLA specifications, by external trusted auditors. The detection of any unexpected behaviour should provide reasoned attribution of faults, trigger alert mechanisms, support notifications and quicker remediation.

4.4.1 Evidence analyser

The method presented for the gathering of evidence is necessarily internal (and specific) to the each service in the cloud ecosystem. The output however shares a similar format and structure, consisted of linked chains of records, assembled to describe events within the service provision, with accountability as a contextual background, supporting demonstration of data processing, correctness of services and business practices.

The events recorded indicate either normal (expected) behaviour or expose particular faults in the service. A verification process becomes necessary to detect and address as soon as possible such faults as policy violations and potential security breaches.

For a validation to be performed, elements from both records and policies will need to be related and compared. A parsing process will collect the elements expressed in the policies that define the agreed obligations, authorizations and access rules. The appropriate values present in the evidence will be considered, and the actions and operations matched against corresponding policy obligations, i.e., the actions expected and allowed will be compared with the actions observed (recorded). Ambiguous situations or further investigations may require in addiction data mining in the supporting elements, most possibly system logs or contracts specifying permissions and authorizations.

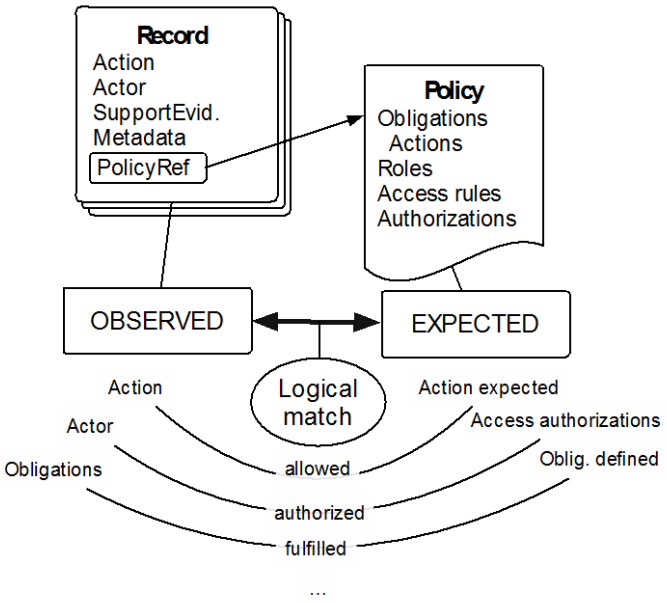


Figure 7: Validation against policies obligations and rules.

Our primary approach will focus on a knowledge base with facts and rules built from the parsed values of applicable policies and the records' elements stated as facts. An associated inference engine may then be used to detect inconsistencies between facts and rules, or infer faults and violations as new facts. The history of previous faulty situations, including false positives, can be of use to support and improve the quality of the validation process with time.

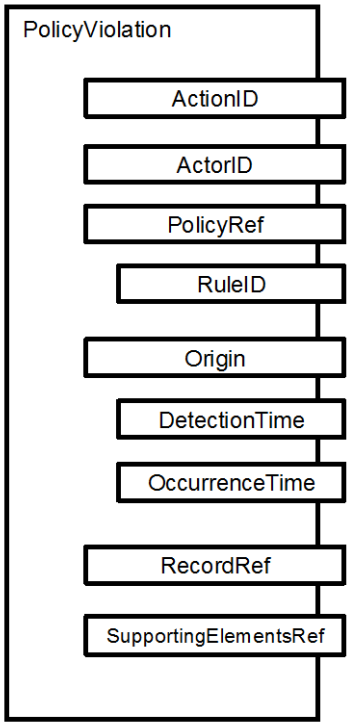
A situation expected to also be covered by the evidence collection, mainly by the automated audit agents system (AAS), is the case of events detected not described by a record or the insufficient information from supporting elements for fulfil the requirements of a record. This implies unexpected events or procedures, incorrect practices, eventually improper or unauthorized access to data, or a questionable possibility of attribution of responsibilities.

They are indicatives of security breaches or mal-functioning of the services, and will require as minimal action, notification of related subjects. In the last case, of service responsibility, either at low-level of incorrect implementation of applications and security mechanisms, or at a higher level, an incorrect practice from part of the services (e.g., accessing data without explicit permission of Data Subjects). The former, being more critical, will require special attention and double checking, searching for further indication of external attacks or malicious internal action, and if a case of a true positive, leading to forensics investigation.

4.4.2 Policy violation format

A policy violation will be reported as an extended version of the record format, containing the elements that acknowledged the violation and a reference to the faulty action's record. The reference to the record allows the identification of the record's chain associated to that particular resource and respectively access to its events' past history.

Upon detection, the automated audit system will adopt actions according to definitions of the service, taking care of the presentation aspects of the violation evidence (necessarily different depending on the target and objective) and trigger alerts to notification or redress mechanisms. The structure of policies violation format is shown in Figure 8, and the planned language for implementations is XML.



ActionID (Ref): records are planned to have a descriptor of the actions and operations (possibly matching the action-ID of A-PPL)

ActorID (Ref): All actions must be performed by authenticated actors or any component in the service, so their ID must be available any time (assures attribution)

PolicyRef: policy identifier or reference to the policy that is under violation

RuleID: reference or actual rule that has been violated (assumption: a policy might consist of several rules), some may be violated, some not. (e.g., Access rule)

Origin: origin of policy violation (possible origins: reported by single tools, by multiple agreeing tools, by aggregation agent (CEP)); i.e., which agent/tool reported/detected the violation

DetectionTime: date and time of the detection (presumably the timestamp of last record)

OccurrenceTime: date and time/timeframe of first observation of violation (back-trace in the records' chain; can be used to evaluate services' response)

RecordRef: reference (hash) to a record in evidence repository, the entry point in the record's chain where the violation is detected. Possibly several records may need to be collect from that chain reconstructing the trail up to the occurrence time, at least, to build a complete evidence (by tools like AAS or A-PPLE)

SupportingElements(Ref): reference to elements of evidence (in the record) incl. data and metadata associated. Since this include sensitive data only the hashes, already calculated, are used to refer them (and original documents may be assessed only in sever cases as litigations)

Figure 8: Policy Violation Format

4.5 Integration with A4Cloud Tools and Other Operations

In this section we suggest how other tools can integrate their output in a manner consistent with the proposed framework.

4.5.1 Proofs of retrievability and other formal operations

From the perspective of the organisations, the provision of proofs of retrievability, as of any other formal or logical operations, will be registered as service events, which can act as a sort of receipt of the execution and result of the request. For the case of this specific proof, it allows the cloud provider to demonstrate that personal data it hosts is available and can be retrieved by their owners as accorded by mutual agreements or contractual obligations.

For a cloud provider as a Data Subject (subcontracting storage services from another provider), it can provide assurance of data retrievability and, with cumulative results in time, a basis for comparative analysis of proof request timings as possible indicator of performance degradation from their providers of even a possible (undesired) reallocation of storage in different location.

The records associated to a proof of retrievability - at least two, request and result - will have identical standard format as proposed above. For example, actions may be labelled with (request_*PoR, timeOfRequest*) and (*output_PoR, timeOfResponse*), as actors the applicant identifier, presumably the owner of the data, and the provider for the response, with "POR" as purpose, (*applicantID, POR*), (*providerID, POR*), and so on for the remaining fields. References for the applicable policies related to the data under proof must be included; assuring that proper access to that resource can be verified (an unauthorized entity requesting this type of proof upon some data may be considered a violation, even if the result wouldn't expose the data itself).

4.5.2 The A-PPL engine and DTMT

The A-PPL engine works as a policy enforcement tool and interact directly with the automated audit system, AAS, and the evidence collection. It's output, Action Log can be considered evidence of the

accountability of a cloud service and as such the record collector must assemble records of such action. In the case of this tool, following directly the A-PPL language specifications, the log's parameters match directly the record's format (action, timestamp, subjectID, resourceID, metadata: resource location, security flags), and are directly assembled into a record associated with the respective chain of that resource. Figure 9 illustrates such process.

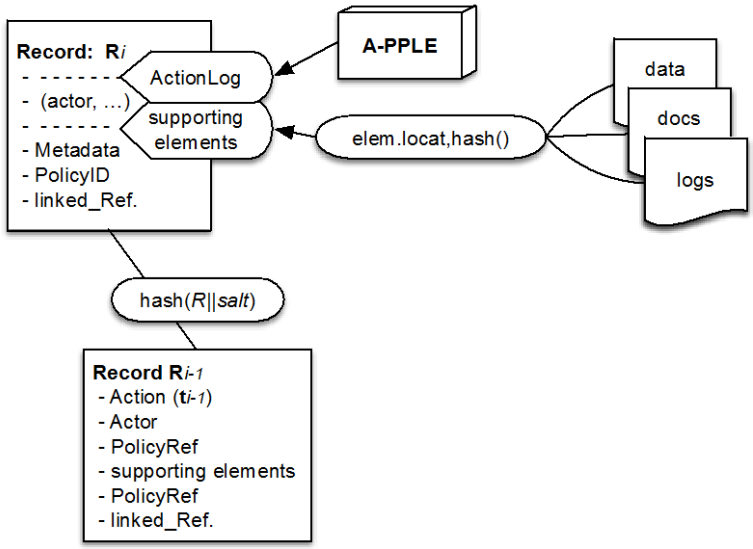


Figure 9: Example of integration of A-PPLE output as evidence element

The Data Transfer Monitor Tool (DTMT) allows the cloud providers to demonstrate compliance with policies concerning personal data transfers. This tool relies on a continuous monitoring of the cloud infrastructure API calls that relate to data transfers, producing logs that can be used to identify potential policy violations. The procedure expected to integration with the framework of evidence is identical to the one with the A-PPL engine, represented in the above scheme. The tool's logs as any other elements in the system utilised for the data monitoring are referenced as supporting elements. The records will naturally be included in the chain of records associated to the processing of a given data, in this case, transfers across different hosts, re-allocation or backups.

4.6 Evidence Repository and Safety Guards

Records will be stored in an evidence repository by the cloud provider; just the small set of the hash references may be made public to the different cloud providers and cloud actors interacting with of a service.

Evidence collected from actions upon data and logs will necessarily constitute big data. We propose the use of available solutions to big data storage, as for example, solutions based on Google's BigTable, which facilitates the use of other big data tools and simplifies distributed big data analytics. Furthermore, most of these solutions include already safety packages, automated data replication and dynamic load balancing for optimal performances, as also backup mechanisms; constituting a good solution for cloud providers, already in possession of knowledge, technology and means.

5 Auditing Process and Evidence

Cloud computing brings along new challenges regarding data security, privacy and auditability. Whereas enterprises were able to quantify their systems security level in non-cloud service provision scenarios this no longer feasible. The lack of control about cloud resources prevents some businesses with rigorous security requirements to move to the cloud. The main disadvantage is the current lack of auditability and certifications when it comes to cloud service provision.

IT security standards[8] describe security controls, which should be implemented in order to be compliant. Furthermore, custom requirements regarding how customer-owned data should be handled in the cloud can be defined. Cloud audits try to provide assurance about the correct implementation of appropriate controls.

The Audit Agent System (AAS) is a software agent-based cloud audit solution, which enables auditors to assure cloud providers are adhering regulatory requirements, information security standards, best practices as well as custom policies. AAS is designed to enable verification of compliance by automating evidence collection, testing procedures and reporting.

5.1 Audit Process

The cloud audit process implemented by AAS comprises of two main processes: a) evidence collection and b) automatic continuous auditing.

a) The evidence collection process builds an information base, which includes required information to conduct audits. This includes the collection of operational evidence (how data is processed in the system demonstrated by logs and other monitoring information), documented evidence (documentation for procedures, standards, policies), configuration evidence (are systems configures as expected), set accountability controls, deployed accountability tools and correct implementation of an accountability process.

Evidence is not collected purposelessly but requires a distinct reason. This reason is defined in an audit policy. Figure 10 depicts the relationship between A-PPL policies, audit policies and audit tasks. A-PPL policies describe access control rules, data usage rules and other obligations related to accountability in a machine-readable way. This is taken as a basis to create audit policies by extracting and mapping these obligations to audit tasks and tools respectively. Since this will be a in some cases a semi-automatic process, input is needed from the auditor (e.g., tool configuration parameters). An audit policy is therefore a container, which includes multiple audit tasks needed to audit the adherence or violation of an A-PPL policy.

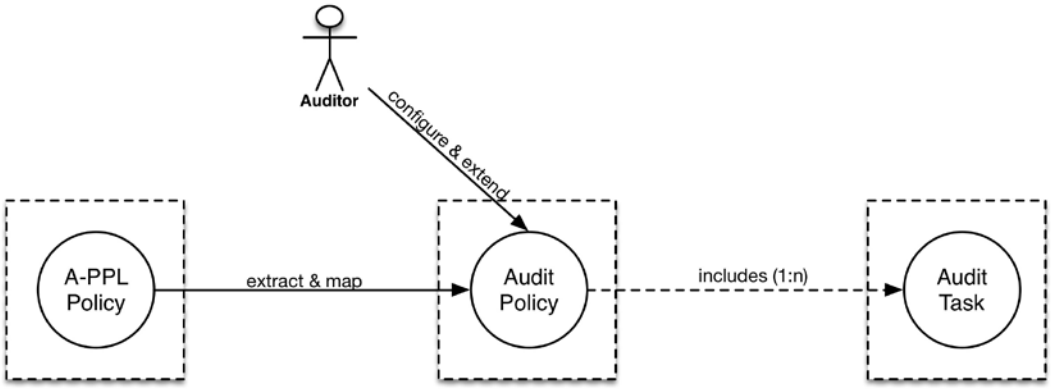


Figure 10: Relationship between A-PPL policies, Audit Policies and Audit Tasks

Whenever possible, this is to be automated by AAS. However, we acknowledge that A-PPL policies will most likely not include information about where required evidence for (non-) compliance may be found, this information has to be complemented by the auditor. This especially includes agent-related information like the mapping of certain kinds of policies to the evidence source that may be needed.

b) Audits in general can be performed periodically or on-demand/continuously, when it is needed (e.g., change in the infrastructure). The main problem of periodical audits in cloud computing is the dynamic change of the infrastructure and therefore, the risk of missing critical accountability issues if the interval

[8] Such as PCI-DSS, COBIT, ISO/IEC 27001, FedRAMP, HIPAA, CSA CCM or NIST SP800-53A.

is too big or if the interval is too short. Hence, a vast number of data from which evidence has to be derived may need to be analysed.

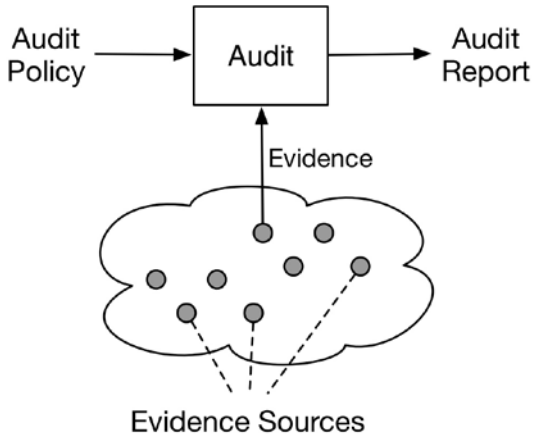


Figure 11: Audit Process

With respect to cloud audits, we assume the following audit process:

1. Planning Phase: Given policies define the automatic audit plan, stated by audit tasks to be processed, i.e. which evidence has to be collected.
2. Securing Phase: Install evidence collection for audit trail collection.
3. Analysis Phase: Automatic evaluation of the collected evidence according to the defined policies, which results in a statement about (non-)compliance with supporting evidence for that claim.
4. Presentation Phase: Presentation in an Audit Dashboard and/or generation of a human-readable document, which includes all processed audit tasks including their results.

Figure 11 depicts these different phases of auditing. Audit policy definitions are an input to the audit process, which analysis evidence collected from various sources in the cloud (the second input). As a result, an audit report is generated, which can take the form of a web-based dashboard presenting policy violations or a notification of other components about policy compliance and violations.

With respect to the AAS, audit agents are used to collect evidence from the various sources of evidence. The collected information is then stored inside the evidence repository. The transformation of the various data formats, which are used in the collection process, (i.e., the various types of agents will interface differently and handle data differently depending of the evidence source) is done by the audit agents. The evidence input format (e.g., XML messages when interfacing with CMSs, syslog format when interfacing with syslogd) as well as the output format required by the evidence repository (more precisely: the format of an evidence record) is known by the audit agents. During the evidence transformation, the format is normalized and (where appropriate) pre-processed.

5.2 Relation to the Framework of Evidence

The framework of evidence forms a conceptual basis for the AAS by providing a defined process for the processing of evidence and a well-defined format for storing evidence data for further analysis. A key component of the framework of evidence is a comprehensive monitoring system, which addresses the various points in a cloud ecosystem, where evidential data is generated. The audit agents described previously represent an actual implementation of this monitoring system. However, it is not the agents themselves that monitor the cloud but they serve merely as a means to interface with various monitoring systems, which are already in place (such as the CMS, monitoring solutions, logs etc.).

5.3 Presentation and Audit Reports

As previously mentioned, audit reports can take different forms of presentation. The most common format is a document that includes the audit results. This kind of documents can be (to some degree)

generated automatically. These types of reports are commonly used when periodic audits are performed. Another way of presenting audit results is a web-based dashboard, which is a more suitable approach when the results of continuous audits need to be presented.

In any case, privacy concerns need to be considered, since audit reports may include sensitive information, which is made available to an external party (in the case of an external auditor). Therefore, it is necessary to limit the amount of evidence presented in an audit report and employ authentication and authorization mechanisms, such as contracts but also technical systems.

5.4 Identification of sources of evidence elements

In this section, we describe sources of data, which can be used as evidence. We thereby focus on a technical point of view on evidence. This means, usability of such data in court is not the main focus of this section. The views on evidence sources in cloud computing presented in this section are based on previous work (Ruebsamen and Reich 2013).

In a cloud environment, there are many sources of evidence. Data collected by logging systems is perhaps the most important type of data, from which evidence may be derived. In cloud environments, logging is performed on several architectural layers. For example, regarding the processing of data in the cloud, provenance information is of high importance. Logging where data originates, how, where and by whom it was processed by maintaining a history of operations in an object's metadata is invaluable information when auditing against data processing policies. Additionally, location information (i.e., where data is and was stored) is gaining importance.

In the following, we describe the different architectural layers, where evidence data is generated and can be collected. These layers are depicted in Figure 12: the network, hardware, host operating system, hypervisor, virtual machines and cloud management system (CMS) layers.

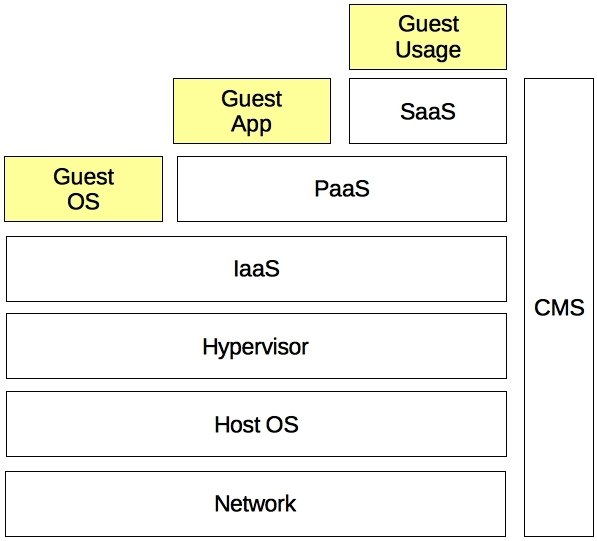


Figure 12: Evidence Sources by Cloud Layers

Network

In a complex computing model, such as cloud, several stakeholders are involved. It needs to be possible to monitor networking resources, which are utilized by a particular tenant. Networking resources can be either physical or virtual and shared among tenants. For instance in IaaS, a single network card in the host machine is utilized by several virtual machines and they may belong to different customers, which is typical in a multi-tenant cloud environment. Isolation between tenants is achieved by virtualizing network resources (e.g., network interface controllers, switches, VLANs etc.). Collecting information about traffic flows can be important to detect data leakage and security policy violations. This is especially important for cloud audits where traffic flow information is important to reveal violations of security policies (e.g., compromised hosts, traffic flows to disallowed locations). Logging of traffic and

communication endpoints (at least on a metadata level where communication endpoints, duration and date are recorded), can therefore be an important source of evidence, especially in IaaS cloud deployments.

Host Operating System
Evidence collected in the host operating system level (i.e., servers hosting virtual machines) comprises of load information (e.g., CPU, RAM utilization), performance counters (e.g., network and I/O counters) and various information collected by monitoring tools (e.g., process monitoring).

Hypervisor
The hypervisor is used to operate virtual machines. It has full control over virtual machines and assigns resources to them. It can provide runtime statistics, but also information derived using advanced techniques like Virtual Machine Introspection (VMI) (Garfinkel, Rosenblum, and others 2003). Evidence collected from the hypervisor can be invaluable, since a VM can be observed from the outside, without interfering with its operations. This allows in-depth analysis of the activities inside the VM, while being undetectable, for instance during an online-forensic analysis.

IaaS
At this level, one of the most interesting evidence acquisition techniques is virtual machine (VM) snapshotting. At any time, a copy of a VM can be created to preserve the actual state of a VM. Beside the VM, all cloud resources (e.g., storage, network, etc.) are important sources of evidence. Additionally, runtime information such as previously described for the Host Operating System layer, might also be collected inside the VM. This, however, implies the usage of some sort of collection software, which takes into account the dynamics of IaaS (e.g., rapid elasticity of cloud resource provisioning). Since virtualized resources in an IaaS scenario are typically under the administrative control of the cloud customer, the trustworthiness of information collected on this layer can quickly become an issue.

PaaS
In a web service PaaS scenario evidence collection can be performed by the cloud provider as well as by the cloud customer (i.e., platform developer). On the provider side, runtime environment logging (e.g., webserver, java runtime, database service, access control service) is of utmost importance. The major concern regarding information collection on this layer evolves around the segregation of multi-tenant log information at the provider-side. Additionally, the customer is free to implement its own evidence collection mechanisms (e.g., application logging). However, to what extent this information can be trusted is beyond the scope of this work.

SaaS
On the SaaS level evidence may come from audit and logging APIs provided by the SaaS service provider. Such APIs may also include authentication and access records. However, the actual content of such data is highly dependent on the cloud provider and application type. Another source of evidence is transient data stored on the client side. Evidence collection in a SaaS scenario might need information from the client side logging (e.g., javascript logging of operations) and other data stored in the browser's cache.

CMS
The Cloud Management System (CMS) is a huge source for evidence information. It is the central controlling component of a cloud infrastructure and provides information about user logins, cloud service usage, access rights, configuration, resource provisioning, policies, location etc. Typically, cloud management systems already provide more or less extensive logging capabilities to monitor the cloud operations (e.g., provisioning of resources, actions performed by cloud customers).

Inter-Cloud
A service provision scenario with multiple cloud service providers demands a correlation of collected evidence from all involved providers. This quickly becomes necessary, when more complex service provision scenarios are considered. For instance, a service provider might choose to provide its software on top of an IaaS provider’s cloud infrastructure. In this case different layers, owned by companies need to be used as evidence sources.

6 Business Use Case: Health Care Services in Cloud

6.1 Brief Description

We use for the exemplification of the framework of evidence, the first use case presented and detailed in WP B-3: HealthCare services in Cloud. This use case deals with the flow of data from cloud customers within the health care domain., where information processed concerns to the heath status of the Data Subjects (the patients). Data is collected in an automated way from sensors (at patients' houses) or uploaded by their respective physicians (medical records, medication history, etc.), eventually accessible by the patients and their relatives/friends (acting in this case as also Data Controllers). The information to be processed and analysed by physicians and caregivers (Hospital) will be stored in an external cloud service.. Naturally, data in any scenario within this Use Case is highly sensitive, concerning personal data directly related with the intimacy and private lives of the Data Subjects, handled and potentially accessible by different services and entities, and as such, of the special care from the perspective of confidentiality and privacy.

The composition of the cloud service is relatively complex, with several cloud providers interacting as both providers and costumers, of each other and a common platform, platform M, a cloud service itself, acting as coordinator service and access point through a graphical interface (see Figure 13).

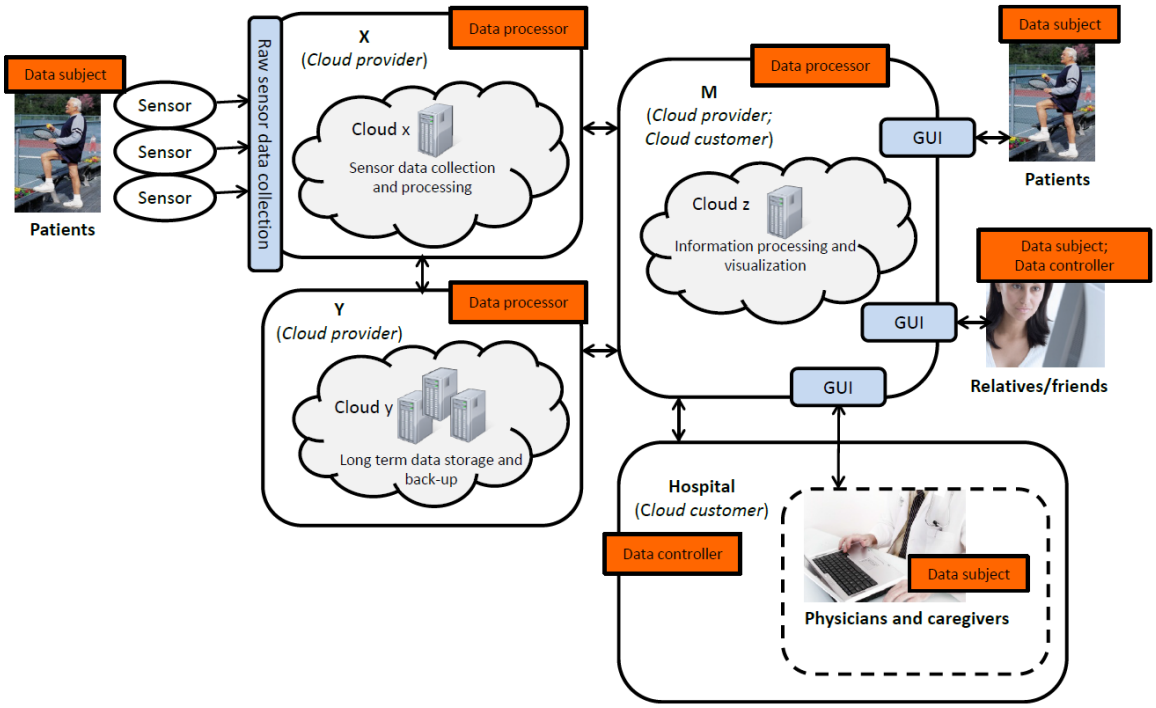


Figure 13: BUC1, healthcare services in the cloud.

The identified actors and the roles in terms of data protection is defined as it follows. Data collected from sensors is processed by cloud X, with long-term data storage and back-ups assured by a second cloud service, cloud Y. Cloud Z processes information and visualization through platform M. All these services are considered Data processors. The hospital is considered a Data Controller although its personnel (physicians and caregivers) are seen as Data Subjects, along with the patients and their relatives and friends. The provision of data, from sensors to patients access, for example, can be seen as a Service Delivery Chain as we described before, were the Data Subject interact with platform M (a PSP) without being aware of services in the chain as cloud X or cloud Y. More details on the complexity of this Use Case, their relations as actors, obligations from different perspectives, etc., can be found in the deliverables DB-3.1, Use Case Descriptions (Bernsmed et al. 2013) and above referenced internal report MSB-3.1, of WP B-3.

6.2 Obligations and Potential Violations

We have identified in the previous Milestone Report MS C8.1, a number of obligations for this use case, which need to be handled in the accountability framework. Here we list the most important obligations. The list of obligations have been extracted from the list of accountability relationships for the use case 1, which is documented in the deliverable DB-3.1.

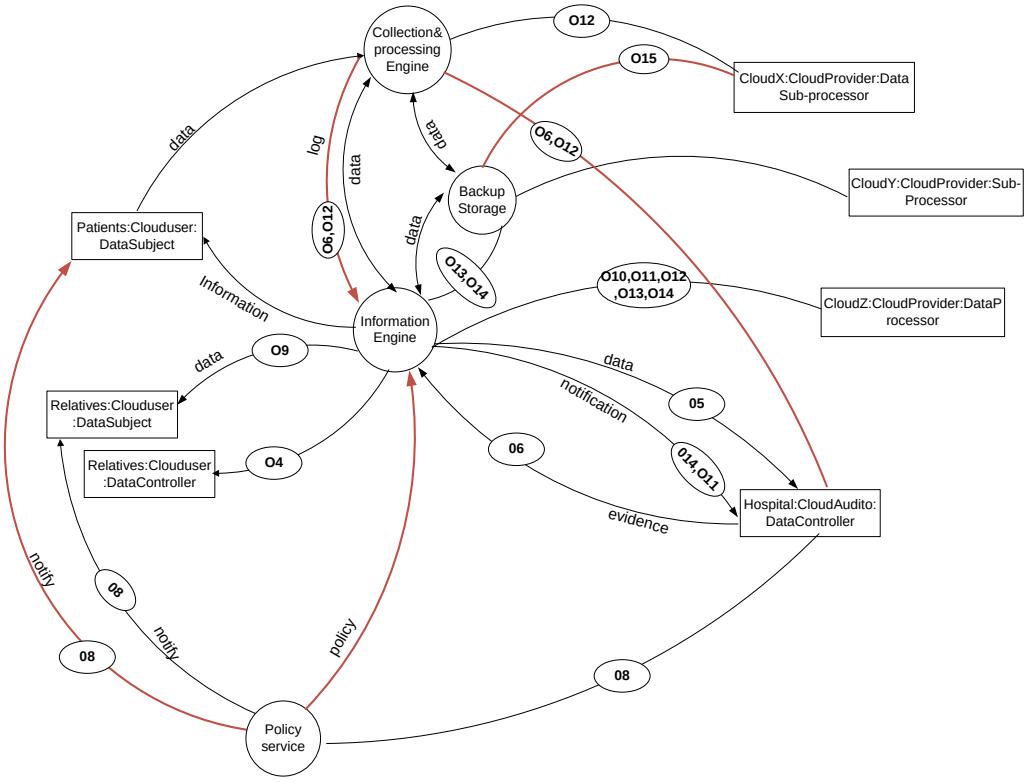


Figure 14: Obligation View for BUC-1 Health Care.

Obligations:
Note: Obligations listed are directly based on the Health Care use case and as such present a less general formulation, that can in most cases be applicable to any general scenario (in bold in this text).

O1: As **data subjects**, the patients **have the right to access, correct and/or delete their personal data**.

A-PPL:
Read and write access control is achieved through XACML rules (See also Appendix, A2).
Deletion is expressed as a Trigger/Action (TriggerAtTime/ActionDeletion)

Violations:
- Data access denied/not allowed or incomplete (e.g. read-only) to the resource-owner, the Data Subject. (Such denial, violating this obligation, may also indicate conflicting policies or incorrect access/authorizations configurations at another point of the Cloud Service)
- Data inexistent (data loss or incorrect stored)
- Deletion not authorized, not executed (data still available) or delayed (case of automatically deletion upon a retention period).

O2-4: Informing about processing, purposes, recipients and rights
(see bellow sub-sections for more details on these obligations)

Violations:
- Not informing.
- (Too) Delayed notification
- Incorrect Recipient (Privacy violation)
- Insufficient or incorrect information

The next two obligations are directly related to a policy Enforcement engine (A-PPLE):

O5: As a **data controller**, the hospital **must use personal data for the specified purposes** only.

O6: As a **data controller**, the hospital **must, upon request, provide evidence to the data subjects** (the patients) **on their personal data processing practices**.

A-PPL:
Purpose(s) is as an attribute expressible in an A-PPL policy.
Request Evidence is a action defined in A-PPL, ActionEvidenceCollection action.

Violations:
- Purpose inadequate or unidentifiable upon use of personal data and agreement.
- Recorded use of personal data not matching the purpose(s) allowed.
- Data processed by entity in groups not specified or denied by the policy (e.g data improperly used for marketing or research purposes)
- Evidence not available/insufficient, upon request.

Obligations related with notifications:

O7: As **a data controller**, the hospital **must notify the processing of personal data to the competent Data Protection Authority**.
It's a particular case of obligation on notification. Violations will be verified in the same manner and contemplate the same types as above.

O8: As a **data controller**, the hospital **must notify the data subjects** (i.e. the patients) **of security or personal data breaches**.
This obligation (action) is designed as a reaction upon detection of such cases (triggers TriggerOnPolicyViolation or TriggerOnDataLost) i.e., after the detection of the mentioned violations or critical faulty behaviour.

Violations:
- Registered policy violation without a registered (record) notification.
- Verified inexistence of data (lost or corrupted) without a registered (record) notification.
- Data improperly accessed without a registered (record) notification.
- (Too) delayed notification

O13: As a **data processor**, the MedNet platform provider **must, upon request, provide evidence to the data controller** (ie. the hospital) **on the correct and timely deletion of personal data**.

Note: same case as O1, but between two cloud providers in a chain and with a time conditioning.

Violations:
- Evidence not provided or incomplete/insufficient.
- Deletion not authorized or not executed (data still available).
- Deletion demonstrated, but after the defined period of time

O15: As **data processors**, the MedNet platform provider, the provider of Cloud x and the provider of Cloud y **must facilitate for regulators to review evidence of the processing of personal data whenever requested**.

Violations:
- Evidence denied or delayed upon request from the regulators

- Detection of Evidence tampered, incomplete or deleted.

6.3 Identifying Elements and Sources of Evidence

In general the above obligations can be grouped in four generic types, which will constitute the decisive factor in the choice of the sources to monitor in the services, and elements to collect for the records' compilation:

i. Data Access
ii. Processing Practices
    a. Data Deletion
iii. Notification
iv. Evidence Provision and Availability

As a title of example, a system's administrator may make the following decisions:

Type of Evidence	Supporting element	Source of Evidence
Data Access	Files' metadata IDs from logs Service API logging PoR	System, platform Authentication Mechanism(s) Service API Cryptographic algorithm/procedure
Processing practices Data deletion	Files metadata Logs (PoR)	Tools, applications Service API, System
Notifications	Logs, Mails Digital copies of correspondence	Mail Server Backup Server
Evidence Provision and availability	Record's chains Audit reports	Evidence repository Audit System

Table 2: Elements and Sources of Evidence

Those supporting elements will be referenced (by name/location) in the records, and only their hashes recorded. As it was stated, they will only be exposed to the proper entities (as data owners relatively to its data, or the recipient of a mail in relation to its correspondence) upon request. Evidence is primarily analysed by auditors at level of actions and operations registered in the service and its comparison against the applicable policies.

6.4 Policy Expression and Violations Detection

Here we give an example of some of those obligations specified in policies with expression in A-PPL format, and consider potential elements to collect and compare from a record chain. We consider for this example a sample of the policies from WP C-4 deliverable, Policy Representation and Enforcement Techniques (Onen et al. 2014) , with the following obligations:

- O1-4: Informing about processing, purposes, recipients and rights:

  "The hospital is accountable to patients, relatives/friends and hospital staff for informing that their personal data is being collected, the purpose of the processing, the recipients of their personal data and the rights the data subjects have in relation to the processing of their personal data. To express this obligation with A-PPL language, the hospital prepares an A-PPL policy that states the proposed data handling rules."

```
<!-- Notify the data subject about the processing policy -->
<Obligation >
      <TriggerOnDataCollection >
            <MaxDelay >
```

```
                    <!-- Notify within 2 minutes -->
                    <Duration >0Y0M0DT0H2M0S </Duration >
              </MaxDelay >
       </TriggerDataCollection >
       <ActionNotify >
              <Media>e-mail</Media>
              <Address>data.subject@example.com</Address>
              <Recipient>Data Subject</Recipient>
              <Type>Data Handling Policy</Type>
       </ActionNotify >
</Obligation >
```

Possible Violations:
- Not informing.
- (Too) Delayed notification
- Incorrect Recipient (Privacy breach)
- Insufficient or Incorrect information

Elements to collect from the records to compare with the required ones, expressed in the applicable policy:

- Time stamping of notifications
- E-mails sent, records of letters, etc.
- Recipients (for confidentiality breaches verification)

In full extent, the record's attributes, referenced by $(subjectID, ResourceID)$ are:

$[actionID, time, agentID, (purpose), \{log_i, hash(log)_i\}, \{metadata_i\}, \{PolicyRef_i\}, refPrev]$

We assume that, to avoid a massive amount of notifications to the users, there are some service policies to control this number and one can be that notifications are sent once upon first reception of personal data and once at start of reception of automated data streams (by sensors, for example).

For an example we consider the patient Kim, from the first use case; assuming that some service indentifiers are defined:

patient:kim = *userID*: P12345
resource:medicalRecord = *resourceID*: MR00001

After the system receives the first medical record of the patient Kim, the TriggerOnDataCollection will fire the action ActionNotify. A notification to the user Kim referent to the processing of his data, a resource identified the identifier MR00001, will be sent and recorded, with the rest of other events associated to (subjectID, ResourceID) = (P123456, MR00001).

The records chain for (P123456, MR00001) will be like:

Receive, 20140101101010, Cloudz.admin.e987, admin, {/path/to/log,hash(log)},{metadata}, *ref_to_(P123456, MR00001)*

… …

Notify, 20140101101012, Cloudz.admin.e987, admin, {/var/log/maillog, *hash*(maillog)}, {metadata}, *ref_to_*R_{i-1}

… …

At the record compilation, the metadata planned upfront for each case is extracted from the defined sources of evidence (eventually chosen according to the type of evidence). In the case of the last record

in the list, referent to the notification, information is extracted from the mail log (eventually some normalization may be required) and should look like:

Metadata:

LogTime = `20140101101015`
`To = hospital.admin(a)hospital.medcare.com, p12345(a)somemail.com`
Size = `589`
Status = `Sent`

With a set of attributes describing the action (and supported by the low level component in the system), verification against the elements in the policy that reflect expressed obligations can be performed.

A set of logical propositions can be established for each action expected or mapped by the policy. (In these example we do not have any specific language in mind, so implementation is totally open)

NOTIFICATION_SENT = (Status == sent AND size !== 0)

NOTIFICATION_ON_TIME = (Time <= Record.timestamp + Policy.MaxDelay)

NOTIFICATION_TO = (To == Policy.Address AND Policy.Recipient == Subject)

That would imply respectively, if any of its logical values were false, a policy violation. The first case, the notification was not sent (or was empty). The second case a delay was detected (some setting in the system can establish the limit upon which a delay can become critical). The third, an incorrect address/subject, by omission or a non-unique recipient, would indicate a violation of privacy, with confidential data potentially sent to incorrect addresses.

More complex rules may be constructed and verified by gathering more sources. For example, the service's templates for different notifications and the mails itself can be text mined (the service owns the notification mail, not raising privacy issues in this process) for defined tokens that matches the type of notification. With a little more of complexity, allowing to verify if the notification contains the pertinent elements defined by the policy (e. g., informing about purposes of processing or rights of the Data Subject).

A basic set of verifications can also be executed for any type of action, e. g., if the *agentID* belongs to a group authorized to perform the action logged or, if a purpose is specified, if the action from the identified *agentID* is valid for a rule defined for that purpose (e.g. purpose: research, may concede authorized reading-access to some types of data from agents identified with research department, purpose marketing may be specifically denied from some users, etc.)

Attribution can be considered assured since any agent, employee or component of automated process is well identified (authentications procedures are assumed to exist in any cloud service provider). A registered action without an authorized or recognized agent, or any detected change in the data without a mapping of an action-actor by a record must be considered also a policy violation and may be evaluated the situation (for the last case) of a potential security breach, requiring further investigation and a possible case for forensics.

7 Conclusions

This deliverable presents a conceptualization of a Framework of Evidence for accountability of Cloud Services. Its design aims to ensure the gathering of enough, verifiable and trustworthy information on the services, practices, and operations. The framework is oriented to the assessment of accountability for cloud services, as it is being investigated and detailed in the conceptual framework of the A4Cloud project (Felici and Pearson 2013), WP C-2, i.e., providing organisations with support to give account of service obligations fulfilment and policies compliance. Additionally, and still at a conceptual level, it proposes a definition for accountability evidence, based on the above referred approach for accountability of cloud services, and focused on enable cloud providers to demonstrate their practices and gather supportive elements for the account of services.

The framework of evidence is an up-front planned, proactive approach, relying on a continuous monitoring with the record of evidence on an event-base (minimising the amount of data to store) to capture the dynamics of the services' practices and data processing; policy specific, assuming the availability of a machine-readable language to express policies and an automated auditing system.

We provide a method for integrity assurance and verification. Nevertheless, we suggest two alternatives to that approach to keep the evidence collection methodology-agnostic, namely the use of trusted third parties TSA, and the possibility of using the Transparent Log (TL) tool from A4Cloud toolkit to provide secure logging with full privacy awareness.

We opt to propose to register only the hashes of elements of evidence as logs, documentation and data in general, as a basic mechanism of tampering-evidence, simultaneously avoiding the exposition of confidential data and minimize the size of a record to the footprint-size of a few bits from the hash function output. It assumes the responsibility of the providers in keeping those sources, based on the willingness to provide evidence for accountability.

The concrete benefits and advantages of this framework for evidence are:

- Third party audits can be done upon the information collected in the records without immediate access to evidence sources (elements' integrity is assured by the hash references; and in ultimate cases, like litigation or forensics analysis, logs or other elements can be provided with restricted access)
- The standardisation of collected elements of evidence records eases validation of operations against the access rules and obligations expressed in the policies, an in special in scenarios with multiple actors running their own framework implementations. Verification can be done either internally (for routine checking, e.g.) or externally (for auditing processes or data subjects inspection), with a clear form of information's synchronisation and cross-reference.
- The amount of space necessary for evidence storage is limited by the use of hashing and chaining. It is important to note that it is not necessary for auditors to store related logs and data (Cloud provider is responsible that his logs are consistent with provided records).
- The publishing of the signed digested references ensures that data has its origins defined and cannot be tampered unnoticed, fixating logs or any other element of evidence (non-repudiation).
- Events can be back-traceable in time following the chain of evidence records linked by the hash references, allowing auditors and Data Subjects to verify performed operations or simply follow the location of data, using the published hash references' chain (attributability).

No constrictions are put in the information to collect, option that is left open and to de decide by the architects and the administrators of accountable cloud services, based on the particular model of each case. The type of information, and above all the mapping of actors and events with the service to provide, must guide the gathering of evidence for accountability. We anticipate that the framework may be applied in cloud services in general, without a specific model or infrastructure as ideal target.

7.1 Open Issues and Future Work

The major open issues relative to the framework of evidence relates to scalability and expected performance, when the complexity of services (implicit in any real-world situation) or the volume of collected evidences may compromise efficiency. Parallel to those is the question of which potential costs (technology investments, necessary third parties, etc.) the cloud service providers would face with the implementation of the framework.

We proposed a possible solution avoiding the need of a third party (a time stamping authority), replaceable by a common-maintained distributed service of timestamped-hashed references. This approach may reveal of special interest in the context of cloud service providers, due to its simple implementation as a cloud service with expectable low costs for maintenance and storage. However, the proficiency of the violations detection (quality/quantity) and processing efficiency, in particular, the case of complex cloud service delivery chains with intricate services' architecture and share of private data in the processing stage still demands attention and future investigation.

Equally mandatory is a comparative analysis with the current industrial solutions that handle events (as SIEM technologies) and are commonly accepted as standards, including not only evidence provision mechanisms but also tracking and monitoring tools. The evolution of this work must include a survey of those solutions (with special focus in the ones used in distributed environments and cloud services), with identification of gaps and incomplete features for accountability.
Another important related question is how this framework, and possible future implementations, will cope with other services and systems already producing evidence in different ways. How the framework fits such scenario. What it complements and what needs to be additionally considered must be part of the work to be developed in the second iteration of this deliverable.

The DoW specifies for the second iteration of this work, modifications highlighted by the use case instantiation (potentially addressing the points listed here). As such work will also focus on the validation process and in the evidence analyser. Attention must be put in the evidence collectable (mainly by AAS agents) not directly mapped to events (relation to detective approach). Is important to assure that non-compliance by lack of actions is being identically detected as the resultant of wrong or incorrect actions. Equally important is the expected input from WP C-5 in order to cover possible gaps in the collected information for construction of metrics.

8 References

Accorsi, Rafael. 2012. “A Secure Log Architecture to Support Remote Auditing.” *Mathematical and Computer Modelling*. doi:10.1016/j.mcm.2012.06.035.

Ateniese, Giuseppe, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. 2007. “Provable Data Possession at Untrusted Stores.” In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, 598–609. CCS ’07. New York, NY, USA: ACM. doi:10.1145/1315245.1315318.

Bernsmed, Karin, Massimo Felici, Anderson Santana de Oliveira, Jakub Sendor, Nils Brede Moe, Thomas Rubsamen, Vasilis Tountopoulos, and Bushra Hasnain. 2013. *Use Case Descriptions*. Technical Report D:B-3.1. A4Cloud Deliverables. A4Cloud.

Bernsmed, Karin, Massimo Felici, Anderson Santana de Oliveira, Jakub Sendor, and Thomas Rubsamen. 2014. *Uses Cases Available*. A4Cloud Milestone Report (work in progress) MSB-3.1. A4Cloud Reports. A4Cloud.

Blažič, Aleksej Jerman, Tomaž Klobučar, and Borka Džonova Jerman. 2007. “Long-Term Trusted Preservation Service Using Service Interaction Protocol and Evidence Records.” *Computer Standards & Interfaces* 29 (3): 398–412. doi:10.1016/j.csi.2006.06.004.

Blibech, Kaouthar, and Alban Gabillon. 2006. “A New Timestamping Scheme Based on Skip Lists.” In *Computational Science and Its Applications-ICCSA 2006*, 395–405. Springer. http://link.springer.com/chapter/10.1007/11751595_43.

Brandner, Ralf, Ulrich Pordesch, and Tobias Gondrom. 2014. “Evidence Record Syntax (ERS).” Accessed May 11. http://tools.ietf.org/html/rfc4998.

Bs, W. Jerry Chisum. 2011. “Crime Reconstruction and Evidence Dynamics.” In *The Forensic Laboratory Handbook Procedures and Practice*, edited by Ashraf Mozayani and Carla Noziglia, 105–22. Humana Press. http://link.springer.com/chapter/10.1007/978-1-60761-872-0_4.

Buldas, Ahto, Peeter Laud, Helger Lipmaa, and Jan Villemson. 1998. “Time-Stamping with Binary Linking Schemes.” In *In Advances on Cryptology (CRYPTO*, 486–501. Springer-Verlag.

Buldas, Ahto, Helger Lipmaa, and Berry Schoenmakers. 2000. *Optimally Efficient Accountable Time-Stamping*.

Casey, Eoghan. 2002. “Error, Uncertainty, and Loss in Digital Evidence.” *International Journal of Digital Evidence* 1 (2): 1–45.

———. 2011. *Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet*. Academic press. http://books.google.com/books?hl=en&lr=&id=6gCbJ4O4f-IC&oi=fnd&pg=PP2&dq=Digital+evidence+and+computer+crime&ots=Wpx4lzVxh0&sig=657KzswsBv8H9fRU9-CED98zmBo.

Castelluccia, Claude, Peter Druschel, Simone Fischer, Aljosa Pasic, Preneel, and Tschofenig. 2011. “Privacy, Accountability and Trust – Challenges and Opportunities — ENISA”. Report/Study. https://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/pat-study.

Cederquist, J.G., R. Corin, M.A.C. Dekker, S. Etalle, and J.I. Den Hartog. 2005. “An Audit Logic for Accountability.” In , 34–43. Ieee. doi:10.1109/POLICY.2005.5.

Chisum, W. Jerry. 2011. “Crime Reconstruction and Evidence Dynamics.” In *The Forensic Laboratory Handbook Procedures and Practice*, 105–22. Springer. http://link.springer.com/chapter/10.1007/978-1-60761-872-0_4.

Crosby, Scott A., and Dan S. Wallach. 2009. “Efficient Data Structures For Tamper-Evident Logging.” In *USENIX Security Symposium*, 317–34. https://www.usenix.org/event/sec09/tech/full_papers/crosby.pdf.

Dixon, P.D. 2005. “An Overview of Computer Forensics.” *IEEE Potentials* 24 (5): 7–10. doi:10.1109/MP.2005.1594001.

Dwork, Cynthia. 2006. “Differential Privacy.” In *Automata, Languages and Programming*, 1–12. Springer. http://link.springer.com/chapter/10.1007/11787006_1.

Felici, Maximo, and Siani Pearson. 2013. “A4Cloud Project: MSC-2.3 Conceptual Framework”. A4Cloud.

Garfinkel, Tal, Mendel Rosenblum, and others. 2003. “A Virtual Machine Introspection Based Architecture for Intrusion Detection.” In *NDSS*, 3:191–206. http://www.isoc.org/isoc/conferences/ndss/03/proceedings/papers/13.pdf.

Guts, Nataliya, Cédric Fournet, and Francesco Zappa Nardelli. 2009. “Reliable Evidence: Auditability by Typing.” In *Computer Security - ESORICS 2009, 14th European Symposium on Research in Computer Security, Saint-Malo, France, September 21-23, 2009. Proceedings*, edited by Michael Backes and Peng Ning, 5789:168–83. Lecture Notes in Computer Science. Springer. http://dx.doi.org/10.1007/978-3-642-04444-1.

Holt, Jason E. 2006. “Logcrypt: Forward Security and Public Verification for Secure Audit Logs.” In *Proceedings of the 2006 Australasian Workshops on Grid Computing and E-Research-Volume 54*, 203–11. Australian Computer Society, Inc. http://dl.acm.org/citation.cfm?id=1151852.

Juels, Ari, and Burton S. Kaliski,Jr. 2007. “PORs: Proofs of Retrievability for Large Files.” In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, 584–97. Alexandria, Virginia, USA: ACM. doi:10.1145/1315245.1315317.

Kessler, Gary C. 2012. “Advancing the Science of Digital Forensics.” *Computer* 45 (12): 25–27. doi:10.1109/MC.2012.399.

Killalea, Tom, and Dominique Brezinski. 2002. “Guidelines for Evidence Collection and Archiving.” http://tools.ietf.org/html/rfc3227.

Kremer, Steve, Mark Ryan, and Ben Smyth. 2010. *Election Verifiability in Electronic Voting Protocols*. Springer. http://link.springer.com/chapter/10.1007/978-3-642-15497-3_24.

Le Metayer, Daniel, Eduardo Mazza, and ML Marie-Laure Potet. 2010. “Designing Log Architectures for Legal Evidence.” In , 156–65. Ieee. doi:10.1109/SEFM.2010.29.

Mazza, Eduardo, Marie-Laure Potet, and Daniel Le Métayer. 2011. “A Formal Framework for Specifying and Analyzing Logs as Electronic Evidence.” In *Formal Methods: Foundations and Applications*, 194–209. Springer. http://link.springer.com/chapter/10.1007/978-3-642-19829-8_13.

Mercuri, Rebecca. 2002. “A Better Ballot Box?” *Spectrum, IEEE* 39 (10): 46–50.

NIST - Information Technology Laboratory. 2012. “FIPS 180-4, Secure Hash Standard (SHS) - Fips-180-4.pdf.” http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf.

Onen, Melek, and Tobias Pulls. 2013. *Privacy Design Guidelines for Accountability Tools*. Technical Report D:37.2 (D:C-7.2). A4Cloud Deliverables. A4Cloud.

Onen, Melek, Jean-claude Royer, Monir Azraoui, Anderson Santana de Oliveira, and Karin Bernsmed. 2014. *Policy Representation and Enforcement Techniques*. Technical D34.2. A4Cloud Deliverables.

Park, J.S., E. Spetka, H. Rasheed, P. Ratazzi, and K.J. Han. 2012. “Near-Real-Time Cloud Auditing for Rapid Response.” In *2012 26th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, 1252–57. doi:10.1109/WAINA.2012.78.

Pearson, S., V. Tountopoulos, D. Catteddu, M. Sudholt, R. Molva, C. Reich, S. Fischer-Hubner, et al. 2012. “Accountability for Cloud and Other Future Internet Services.” In *2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom)*, 629–32. doi:10.1109/CloudCom.2012.6427512.

Pretschner, Alexander, Florian Schütz, Christian Schaefer, and Thomas Walter. 2009. “Policy Evolution in Distributed Usage Control.” *Electronic Notes in Theoretical Computer Science* 244: 109–23.

Ruebsamen, Thomas, and Christoph Reich. 2013. “Supporting Cloud Accountability by Collecting Evidence Using Audit Agents.” In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, 1:185–90. IEEE. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6753796.

Schatz, Bradley, and Andrew J. Clark. 2006. “An Open Architecture for Digital Evidence Integration.” http://eprints.qut.edu.au/21119/.

Stamm, M., M. Wu, and K. J. R. Liu. 2013. “Information Forensics: An Overview of the First Decade.” *IEEE Access* 1 (1): 167–200.

“Trend Report: Top Trends 2012-2013 - 70516.” 2013. *KuppingerCole*. Accessed August 23. http://www.kuppingercole.com/report/trendreporttop2012200412.

Turner, Philip. 2005. “Unification of Digital Evidence from Disparate Sources (digital Evidence Bags).” *Digital Investigation* 2 (3): 223–28.

Vaughan, Jeffrey A., Limin Jia, Karl Mazurak, and Steve Zdancewic. 2008. “Evidence-Based Audit.” *2012 IEEE 25th Computer Security Foundations Symposium* 0: 177–91. doi:http://doi.ieeecomputersociety.org/10.1109/CSF.2008.24.

Volonino, Linda. 2003. “Electronic Evidence and Computer Forensics.” *Communications of the Association for Information Systems* 12. http://search.ebscohost.com/login.aspx?direct=true&profile=ehost&scope=site&authtype=crawler&jrnl=15293181&AN=11386512&h=CYisguZlEZv4L3hwMwagLWdoAQfEdakJWSJCt0nxfXvrcaIMLCZpBZBDabNklR96T2EdEpKfJa068aAtY0wbLA%3D%3D&crl=c.

Wang, Chen, and Ying Zhou. 2010. “A Collaborative Monitoring Mechanism for Making a Multitenant Platform Accountable.” *Proc. HotCloud*. https://www.usenix.org/legacy/event/hotcloud10/tech/full_papers/WangC.pdf.

Yao, Jinhui, Shiping Chen, Chen Wang, D. Levy, and J. Zic. 2010a. “Accountability as a Service for the Cloud.” In *2010 IEEE International Conference on Services Computing (SCC)*, 81–88. doi:10.1109/SCC.2010.83.

Yao, Jinhui, Shiping Chen, Chen Wang, David Levy, and John Zic. 2010b. “Accountability as a Service for the Cloud.” In *Services Computing (SCC), 2010 IEEE International Conference on*, 81–88. IEEE. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5557218.

Yumerefendi, Aydan R., and Jeffrey S. Chase. 2007. “Strong Accountability for Network Storage.” *ACM Transactions on Storage (TOS)* 3 (3): 11.

Zhifeng Xiao, Nandhakumar Kathiresshan, Yang Xiao. 2012. “A Survey of Accountability in Computer Networks and Distributed Systems.” *Security and Communication Networks*. http://dx.doi.org/10.1002/sec.574.

9 Appendix

A.1 Tamper-evident chain of records in evidence framework

Consider a chain of records documenting some event supported by a log (or any other element) L.

For any instant i, the log will be in the state L_i. By state, it’s understood that the quantity of material contained in the log includes the most up-to-date information, as also all past history related to the event. The record of the event for that time instant will depend on L_i through its hash value:

$$R_i = R(h_i)\ ,\ \text{with}\ \ h_i = hash(L_i)$$

Upon the compilation of R_i, its hash is calculated (containing the hash of the log) and published in an external service/server:

$$H_i = \big(hash(H_{i-1}), hash(R_i)\big) \ \rightarrow \ public$$

The past history related to that event and changes reflected in the log are then described by the chain of records (internal) and mirrored by its respective hash chain (external):

$$R(h_1) \leftarrow R(h_2) \leftarrow \cdots \leftarrow R(h_i)$$

$$H_1 \leftarrow H_2 \leftarrow \cdots \leftarrow H_i$$

Assumptions: We defined the hash references repository in a way that any actor (including end-users) in a cloud services chain can read at any time, and only properly authorized actors can have granted write access, with the requirement of mandatory digital signing per operation. Access must be defined as write-once/read-many (no re-write or deletion, except under special controlled situation as removable of archaic records).
It’s assumed that this repository is properly secured, both against attacks as accidents (cutting-edge methods of secure data storage and solutions for replication with several levels of redundancy are available, but its description is outside the scope of this demonstration).

Lemma 1. A record cannot be altered without forthcoming detection.

Let R_j' be an altered record of the instant j (for example a dishonest service tries to discard responsibilities for an action or hide unauthorized access to some data). We consider only the use of non-collision hash functions, and as such will have necessarily:

$$H_j' \neq H_j$$

In order to hide this discrepancy, considering that the hash references are published publically, the malicious actor would need to change the hash reference H_j in the external repository.

That procedure would face several problems. Firstly, the act of publish a new reference would need to overpass detection by the monitoring system, which would ordinarily record such action in a new record (documenting the violation!). Secondly, it demands that the discrepancy would meanwhile go undetected by automated routine checks and audit processes.

And moreover, since the hash references are linked in a chain, for each $k \geq j+1$, any H_k will be function of previous H (as hashes of hashes):

$$H_k = H\big(hash(H_{k-1}), hash(R_k)\big)$$

i.e., H is a function of both the record R_k and the hash reference of the previous record:

$$H_k = H(H_{k-1}, R_k)$$

and successively:

$$H_k = H(H_{k-1}, R_k) = H(H(H_{k-2}, R_{k-1}), R_k) = \cdots$$
$$\ldots = H\big(\ldots H\big(H_j, R_{j+1}\big), \ldots, R_k\big)$$

As such, any modification of H_j would also require (to be undetectable):

i) the modification of all R_k accordingly, internally,
ii) the calculation of all new adulterated H_k'
iii) untraceable and undetectable update the news $H_k'$to the external repository.

Since all hash references are accessible publically by any cloud actor and subject of continuous automated audit processes, such complex procedure in two different services and different physical resources couldn't be done without detection, besides that would collide with the restriction of limited write-access.

The issue is then reduced to adulteration of the last record, especially between the phase of record compilation and upload of hash references. For a last record in a chain there is no follow chain elements and the exposed above don't hold.

However, we consider in the fact that this is an automated procedure with multiple instantiation running simultaneously for each CSP. A human processing would be totally unfeasible, and so we rely on the consideration that any software implementing this method would be built independently, without the knowledge of the services to be served, preventing implementations of backdoor accesses or any malicious procedures. As in any software to be deployed in high sensitive services (banking, security), is expected a rigorous verification, performed by specialized trusted 3rd parties, which certify this type of applications.

In order to overcome the automation process, an infractor would need either to crack the existent software or intercept and replace the upload of the hash reference, without being noticed. Although possible, these are criminal acts and they fall out of the scope of accountability validation, to be part of the security threats that any organization face and for each specific methods and analysis in the field of forensics science exist.

We will assume that the complexities behind of such action, plus the requirement of signing the hash references (identifying the committer) shall be enough to prevent attempts of this particular case of record corruption.

Lemma 2. A log or any supporting element of evidence cannot be changed without detection.

Consider an *a-posterior* mischievous alteration in the log entry of instant j:

$$L_j \rightarrow {L_j}^*$$

Since the hash function considered are collision-resistant:

$$hash(L_j) \neq hash({L_j}^*)$$

implying that:

$$R({L_j}^*) \neq R(L_j)$$

An alteration of a log referenced in a record will be easily detectable by comparison with the information contained in the record.

This implies that to make the change consistent, the record needs to be recompiled and its hash reference updated. But that would result identically in a detectable change, according to the previous lemma. The situation of alteration of a record at the moment of the record compilation falls under the same considerations as exposed in the conclusion of lemma 1.

Conclusions. Lemma 1 and lemma 2 show that under usual conditions evidence collection cannot be tampered without notice. Tampering evidence will require extreme methods (e.g. adulterate software) and considerable effort (unauthorized access in two different services/servers), affecting at most the last event in a recording chain.

A.2 Examples of Access rules in A-PPL.

Rules (XACML): PERMIT, DENY, AND:

Data/resources access grants and logical combinations of rules.

Structure:

```
RuleID
Effect
Subject     (SubjectMatch/AttributeValue)
Resource    (ResourceMatch/AttributeValue)
            (ResourceMatch/AttributeDesignator)
Action(ActionMatch/AttributeValue)
```

Syntax:
Permission* Subject Action Resource:

```
Effect Subject Action Resource
```

Ex:

```
Permit admin read database
```

*may be a complex combination of logical conditions (defined also as rules)

Possible access violations:
- Non-allowed Action(s) performed on the Resource(s)
- Not allowed Subject for the required or performed Action
- Specify authorizations on the policy rules may conflict with more generic access authorizations (conflicting/inconsistent policies).