A Cloud Accountability Policy Representation Framework

Walid Benghabrit[1], Hervé Grall[1], Jean-Claude Royer[1], Mohamed Sellami[1], Monir Azraoui[2], Kaoutar Elkhiyaoui[2], Melek Önen[2], Anderson Santana De Oliveira[3] and Karin Bernsmed[4]

[1]*Mines Nantes, 5 rue A. Kastler, F-44307 Nantes, France*

[2]*EURECOM, Les Templiers, 450 Route des Chappes F - 06410 Biot Sophia Antipolis, France*

[3]*SAP Labs France, 805 avenue du Dr Donat Font de l'Orme F - 06250 Mougins Sophia Antipolis, France*

[4]*SINTEF ICT, Strindveien 4, NO-7465 Trondheim, Norway*

{*firstname.lastname*}*@mines-nantes.fr,* {*firstname.lastname*}*@eurecom.fr, anderson.santana.de.oliveira@sap.com, Karin.Bernsmed@sintef.no*

Keywords: Accountability, Data Protection, Framework, Policy Language, Policy Enforcement.

Abstract: Nowadays we are witnessing the democratization of cloud services. As a result, more and more end-users (individuals and businesses) are using these services for achieving their electronic transactions (shopping, administrative procedures, B2B transactions, etc.). In such scenarios, personal data is generally flowed between several entities and end-users need (i) to be aware of the management, processing, storage and retention of personal data, and (ii) to have necessary means to hold service providers accountable for the usage of their data. In fact, dealing with personal data raises several privacy and accountability issues that must be considered before to promote the use of cloud services. In this paper, we propose a framework for the representation of cloud accountability policies. Such policies offer to end-users a clear view of the privacy and accountability obligations asserted by the entities they interact with, as well as means to represent their preferences. This framework comes with two novel accountability policy languages; an abstract one, which is devoted for the representation of preferences/obligations in an human readable fashion, a concrete one for the mapping to concrete enforceable policies. We motivate our solution with concrete use case scenarios.

1 INTRODUCTION

According to (Pearson et al., 2012), accountability regards the data stewardship regime in which organizations that are entrusted with personal and business confidential data are responsible and liable for processing, sharing, storing and otherwise using the data according to contractual and legal constraints from the time it is collected until when the data is destroyed (including onward transfers to third parties). Obligations associated to such responsibilities can be expressed in an accountability policy, which is a set of rules that defines the conditions under which an accountable entity must operate.

Today, there is neither an established standard for expressing accountability policies nor a well defined way to enforce these policies. Since cloud services often combine infrastructure, platform and software applications to aggregate value and propose new cloud applications to individuals and organizations, it is fundamental for an accountability policy framework to enable "chains of accountability" across cloud services addressing regulatory, contractual, security and privacy concerns.

In the context of the EU FP7 A4Cloud project[1] we are currently working on defining a framework where accountability policies will be enforceable across the cloud service provision chain by means of accountability services and tools. Accountable organizations will make use of these services to ensure that obligations to protect personal data and data subjects' rights[2] are observed by all who store and process the data, irrespective of where that processing occurs. Under the perspective of the concept of accountability, we have elicited the following types of accountability obligations that must be considered while designing our policy framework:

- Access and Usage Control rules - express which rights should be granted or revoked regarding the

[1]The Cloud Accountability Project: http://www.a4cloud.eu/.

[2]This work mainly focus on the European Data Protection directive (Directive, E. U., 1995).

use and the distribution of data in cloud infrastructures, and support the definition of roles as specified in the Data Protection Directive, e.g. data controller and data processor.

- Capturing privacy preferences and consent - to express user preferences about the usage of their personal data, to whom data can be released, and under which conditions.
- Data Retention Periods - to express time constraints about personal data collection.
- Controlling Data Location and Transfer - clear whereabouts of location depending on the type of data stored and on the industry sector processing the data (subject to specific regulations) must be provided. Accountability policies for cloud services need to be able to express rules about data localization, such that accountable services can signal where the data centers hosting them are located. Here we consider strong policy binding mechanisms to attach policies to data.
- Auditability - Policies must describe the clauses in a way that actions taken upon enforcing the policy can be audited in order to ensure that the policy was adhered to. The accountability policy language must specify which events have to be audited and what information related to the audited event have to be considered.
- Reporting and notifications - to allow cloud providers to notify end-users and cloud customers in case of policy violation or incidents for instance.
- Redress - express recommendations for redress in the policy in order to set right what was wrong and what made a failure occur.

In this paper we provide a cloud accountability policy representation framework designed while considering the aforementioned requirements. We define an abstract yet readable language, called AAL, for accountability obligations representation in a human readable fashion. We also define a concrete policy enforcement language, called A-PPL, as an extension of the PPL (Ardagna et al., 2009) language. The proposed framework, offers the means for a translation from abstract obligations expressed in AAL to concrete policies in A-PPL.

The rest of this paper is organized as follows. Section 2 describes related work. Section 3 gives an overview on the main components of our policy representation framework. We present the abstract accountability policy language we propose in Section 4 and the concrete one in Section 5. Section 6 describes a realistic use case as a proof of concept to our work. Section 7 discusses our work and presents directions for future work.

2 RELATED WORK

In the following, we provide an overview of related work in the field. We organize this section along the following categories that relate to our contribution in this paper: accountability in computer science, obligations in legal texts and directives, enforcement and policy languages.

2.1 Accountability

There is a recent interest and active research for accountability which overlap several domains like security (Weitzner et al., 2008; Zhifeng Xiao, 2012; Pearson and Wainwright, 2013), language representation (Métayer, 2009; DeYoung et al., 2010), auditing systems (Feigenbaum et al., 2012; Jagadeesan et al., 2009), evidence collection (Sundareswaran et al., 2012; Haeberlen et al., 2010) and so on. However, only few of them consider an interdisciplinary view of accountability taking into account legal and business aspects. We particularly emphasize the work from (Feigenbaum et al., 2012) and (Pearson and Wainwright, 2013) since they provide a general, concrete view and yet an operational approach.

Regarding tool supports and frameworks we can find several proposals (Wei et al., 2009; Haeberlen et al., 2010; Zou et al., 2010), but none of them provides a holistic approach for accountability in the cloud, from end-user understandable sentences to concrete machine-readable representations. In (Sundareswaran et al., 2012), authors propose an end-to-end decentralized accountability framework to keep track of the usage of the data in the cloud. They suggest an object-centered approach that packs the logging mechanism together with users' data and policies.

2.2 Obligations in Regulations

There is an international trend in protecting data, for instance in Europe with Directive 95/46/EC (Directive, E. U., 1995), the HIPAA rules (US Congress, 2002) in the USA and the FIPPA act (Legislative Assembly of Ontario, 1988) in Canada. As an example, Directive 95/46/EC states rules to protect personal data in case of processing or transferring data to other countries. There exist some attempts to formalize or to give rigorous analyses of this kind of rules. In (Métayer, 2009) the authors present a restricted

natural language SIMPL (SIMple Privacy Language) to express privacy requirements and commitments. In (Breaux and Anton, 2005) the authors describe a general process for developing semantic models from privacy policy goals mined from policy documents. In (Kerrigan and Law, 2003), the authors develop an approach where contracts are represented by XML documents enriched with logic metadata and assistance with a theorem prover. In (DeYoung et al., 2010) the authors provide a formal language to express privacy laws and a real validation on the HIPAA and GLBA (US Congress, 1999) sets. These works either are not end-to-end proposals, only cover data privacy not accountability or are only formal proposals without an enforcement layer.

2.3 Enforcement and Policies

A number of policy languages have been proposed in recent years for machine-readable policy representation. We reviewed several existing policy languages (see (Garaga et al., 2013) for details), defined either as standards or as academic/industrial proposals.

Existing policy languages have focused on specific obligations. The eXtensible Access Control Markup Language (XACML) (OASIS Standard, 2013) aims at providing a declarative language for access control. The XML-based languages P3P (Platform for Privacy Preferences Project) (Marchiori, 2002), PPL (Primelife Policy Language) (Ardagna et al., 2009) and SecPal4P (Becker et al., 2010) are used to describe privacy policies and data collection policies. SLAng (Lamanna et al., 2003) and ConSpec (Aktug and Naliuka, 2008) are designed to automatize contract negotiations and to monitor the agreed contract statements. However, all these proposals fail to provide elements for specifying accountability-specific obligations such as logging, reporting, audits, evidence collection and redress.

Having identified the limitations of existing languages, we analyzed their extensibility and their suitability to express accountability obligations. Extensible languages such as XACML and PPL appear to be the most suitable. In our work we consider PPL since it provides elements that capture the best accountability obligations (see Section 5). In a nutshell, PPL extends XACML, an XML-based language aimed for access control, to provide an automatic means to define and manage privacy policies. PPL allows expressing data handling policies (on the data controller side) and data handling preferences (on the data subject side) that are evaluated and matched to output a sticky policy that travels with the data downstream. Therefore, PPL specifies statements on access control, authorizations and obligations. In addition, the language provides a way to declare some accountability-specific obligations such as logging and notifications. However, they fail to capture these obligations accurately and may be unpractical when directly used within an accountability policy. Besides, auditing is not part of PPL since its focus is on privacy and not accountability.

3 THE CLOUD ACCOUNTABILITY FRAMEWORK

In this section, we provide an overview of our proposed policy representation framework. Such framework must allow end-users to easily express their accountability obligations and preferences and even be complete and rigorous enough to be run by a policy execution engine. Hence we are faced with the following dilemma: the policy must be written by an end-user, which does not necessarily have skills in a certain policy language and the policy must be machine understandable at the same time. Machine understandable means that sentences can be read, understood and executed by a computer.

In this context, we propose a policy representation framework (see Figure 1) that allows a user, step (1) in Figure 1, to express his accountability needs in a human readable fashion and (2) offers the necessary means to translate them (semi-)automatically[3] to a machine understandable format.

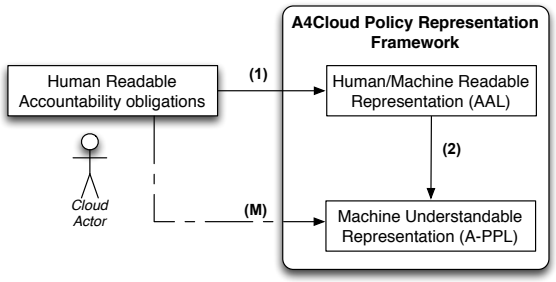

Figure 1: Overview on the accountability policy representation framework.

Accountability as it appears in legal, contractual and normative texts about data privacy make explicit four important roles that we consider in our proposal:

- Data subject: this role represents any end-user which has data privacy concerns, mainly because he outsourced some of its data to a cloud provider.

[3]here "semi" means that sometimes human assistance could be needed.

- Data processor: this role is attributed to any computational agent which processes some personal data. It should act under the control of a data controller.
- Data controller: it is legally responsible to the data subject for any violations of its privacy and to the data protection authority in case of misconduct.
- Auditor: it represents data protection authorities which are in charge of the application of laws and directives.

3.1 Step (1). Human/machine Readable Representation

To express accountability obligations we define an Abstract Accountability Language (AAL), which is devoted to expressing accountability obligations in an unambiguous style and which is close to what the end-user needs and understands. As this is the human readable level, this language should be simple, akin to a natural logic, that is a logic expressed in a subset of a natural language.

For instance, a simple access control obligation to state that "the data d cannot be read by all agents" will be formulated in a human/machine readable fashion using our accountability language as "MUSTNOT ANY:Agent.READ(d:Data)". Details on the AAL syntax are provided in Section 4.

3.2 Step (2). Machine Understandable Representation

In this step (called the mapping), the accountability obligations expressed in AAL are (semi-)automatically translated into a machine understandable policy. We target a policy language that is able to enforce classic security means (like access or usage controls) but also accountability obligations. Such automatic translation may need several passes, due to the high level of abstraction of AAL.

As analyzed in Section 2, the PrimeLife Policy Language (PPL) (Ardagna et al., 2009) seems the most convenient language for privacy policies representation. It can be extended to address specific accountability obligations such as auditability, notification or logging obligations. Hence, we propose an extension to PPL, A-PPL for accountable PPL, which supports such obligations. The details of this extension are described in Section 5.

4 ABSTRACT LANGUAGE

We introduce in this section AAL (Abstract Accountability Language), which is devoted to expressing accountability obligations in an unambiguous human readable style. The AAL concepts are presented in Section 4.1, its syntax in Section 4.2 and we provide an outlook on our approach for a machine understandable representation of AAL policies in Section 4.3.

4.1 AAL Concepts

As explained in (Feigenbaum et al., 2012) an accountable system can be defined with five steps: prevention, detection, evidence collection, judgment and punishment. We follow this line for the foundation of our accountability language. In AAL, usage control expressions represent the preventive description part. Audit expressions encompass the detection, evidence collection and judgment parts. Finally, rectification expressions represent the punishment description part. We use the term rectification since these expressions don't cover only punishment, but also remediation, compensation, sanction and penalty. Thereby, an AAL sentence is a property (more formally a distributed system invariant) expressing usage control, auditing and rectification. The general form of an AAL sentence is: `UsageControl Auditing Rectification` and the informal meaning is: *try to ensure the usage control, in case of an audit, if a violation is observed then the rectification applies*. The reader should also note that there are two flavors of AAL sentences:

- User preferences: expressing the obligations a data subject wants to be satisfied, for instance he does not want its data to be distributed over the network or only used for statistics by a given data processor, and so on.
- Processor obligations (sometimes called obligations or policies): these are the obligations the data processor declares to ensure regarding the data management and processing.

Finally, as many policy representation languages, we consider permission, obligation and prohibition in AAL. They occur in various approaches, like in PPL, or in the ENDORSE project[4]. Permission, obligation and prohibition are respectively expressed in AAL sentences with these keywords: `MAY`, `MUST` and `MUSTNOT`, as advocated by the IETF RFC 2119 (Bradner, 1997).

[4]http://ict-endorse.eu/

4.2 AAL Syntax

Figure 2 shows the syntax of AAL using a Backus-Naur Form (BNF) (Knuth, 1964) like syntax. AAL allows the expression of `Clauses` representing obligations that have to be met either in an accountability policy or preference. A `Clause` has one usage expression and optionally[5] an audit and a rectification expression: `Exp ('AUDITING' Exp)? ('IF_VIOLATED_THEN' Exp)?`. The expression `Exp` of a clause can be either atomic or composite. Composite `Exps` are written in the form `Exp ('OR' | 'AND' | 'ONLYWHEN' | 'THEN') Exp`.

As an example, consider the user preference of a data subject who grants read access to an agent A on its data D. This usage control is a permission, which can be expressed as follows.

```
MAY A.READ(D:Data)
```

But the full accountability sentence does imply that an auditor B will audit the system and, in case of violations, can sanction the data controller C.

```
MAY A.READ(D:Data)
AUDITING MUST B.AUDIT(C.logs)
IF_VIOLATED_THEN MUST B.SANCTION(C)
```

Further examples of user preferences and processor obligations expressed in AAL are provided in Section 6.

4.3 Machine Understandability

Generating machine understandable policies from accountability preferences and obligations written in AAL can be easily done when dealing with usage control clauses. However, this mapping is less obvious for clauses with temporal modalities and with auditing. The main issue for such mapping is the gap between the AAL language, which is property-oriented, and the machine understandable language, which is operational. To fill this gap we need more artifacts, Figure 3 provides an overview on our proposed mapping process.

According to this figure, we can see that going from a human/machine readable representation in AAL to a machine understandable representation of the accountability preferences/obligations (arrow numbered (2) in Figure 3) is done through three steps:

- **(2'.1).** First, a temporal logic is used to make more concrete AAL sentences as temporal logic properties. Indeed, in an accountability policy we should represent the notions of permission, obligation and prohibition. In addition, there is a need to express conditions and various logical combinations. Furthermore, one important thing is to have time, at least logical discrete time, for instance to write: "X writes some data and then stores some logs". Our target is a temporal logic with time, one concrete candidate is mCRL2 (Cranen et al., 2013).
- **(2'.2).** Second, a policy calculus is used to describe the operational semantics associated to the concrete properties defined in (2'.1). This calculus is based on the concept of reference monitor (Schneider, 2000) for both the agents and the data resources. It relies on a previous work for distributed agent communicating via messages (Allam et al., 2012). This operational semantics provides means for abstractly executing the temporal logic expressions. This process is known as "program synthesis", starting from a property it generates a program ensuring/enforcing the property.
- **(2'.3).** Finally, the generated policy using our policy calculus is (semi-)automatically translated to a machine understandable policy based on predefined transformation rules. Our target is the A-PPL extension of PPL which is described in the next section.

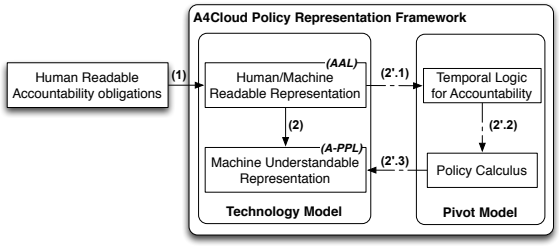

Figure 3: Overview on the machine understandable translation of AAL.

5 CONCRETE LANGUAGE

In this section, we present an enhanced version of PPL with extensions that address the identified limitations of the language to accurately map accountability obligations. We name this accountability policy language (A-PPL).

Our accountability policy representation framework maps AAL clauses to concrete and operational machine understandable policies. As already mentioned in section 2.3, in order not to define yet another completely new language to map accountability obligations to machine-understandable policies, we conducted a preliminary study on existing languages and among all the possible candidates, PPL seems the one

[5] Items followed by ? are optional expressions.

```
(1) Clause ::= (LOGGING? Exp (AUDITING Exp )? (IF_VIOLATED_THEN Exp )?)+
(2) Exp ::= Modal | Exp (OR|AND|ONLYWHEN|THEN) Exp
(3) Modal ::= (MAY|MUST|MUSTNOT) Action
(4) Action ::= Agent . Oper ( Param ) ((BEFORE|AFTER) Time)?
(5) Agent ::= uri
(6) Oper ::= READ|WRITE|LOG|SEND|NOTIFY|service[Agent_provider]
(7) Time ::= date | duration
(8) Param ::= Constant | Variable
(9) Constant ::= stringLiteral
(10) Variable ::= Literal?
```

Figure 2: Excerpt of the AAL Syntax.

that best captures the accountability concepts. Therefore, in this section, we present how A-PPL extends PPL to address accountability obligations.

PPL implicitly identifies three roles: the data subject, data controller and data processor roles. Besides, PPL defines an obligation as a set of triggers and actions. Triggers are events related to the obligation that are filtered by a condition and that trigger the execution of actions. Therefore, PPL defines markups to declare an obligation. Inside the obligation environment, one can specify a set of triggers and their related actions.

5.1 Extension of Roles

To address accountability concerns in a cloud environment, it might be necessary to include in the policy a reference to the role of the subject to which the policy is applied to. For instance, in PPL, it was not possible to identify the Data Controller. We therefore suggest adding to the PPL `<Subject>` element a new attribute, `attribute:role`, for this purpose. Furthermore, in addition to the four roles PPL inherently considers (Data Subject, Data Controller, Downstream Data Controller, Data Processor), A-PPL extends PPL with one additional role. We add the auditor role that is considered as a trusted third party that can conduct independent assessment of cloud services, information systems operations performance and security of the cloud implementation. This new role is important to catch accountability specific obligations such as auditability, reporting notification and remediation.

5.2 Extension of Actions and Triggers

We add to PPL a set of new A-PPL actions and triggers in order to map accountability obligations. In particular, we enhance the action of logging `ActionLog` and notification `ActionNotify` that already exist in PPL. For instance, while PPL currently enables notification thanks to the `ActionNotifyDataSubject`, A-PPL defines a new and more general `ActionNotify` action in which one can define the recipient of the notification thanks to a newly defined parameter `recipient`;moreover, the additional `Notificationtype` parameter defines the purpose of the notification which can be policy violation, evidences or redress for example. On the other hand, the current `ActionLog` action in PPL fails to capture accountability obligations. The new `ActionLog` action in A-PPL introduces many additional parameters to provide more explicit information on the logged event: for example, `timestamp` defines the time of the event, and `Resource location` identifies the resource the action was taken on. We also create two actions related to auditability: `ActionAudit` that creates an evidence request and `ActionEvidenceCollection` that collects requested evidence. In addition, auditability requires the definition of two new triggers related to evidence: `TriggerOnEvidenceRequestReceived` that occurs when an audited receives an evidence request and `TriggerOnEvidenceReceived` that occurs when an auditor receives the requested evidence. Similarly, when an update occurs in a policy or in a user preference, the update may trigger a set of actions to be performed. Thus, we create two additional triggers: `TriggerOnPolicyUpdate` and `TriggerOnPreferenceUpdate`. Finally, to handle complaints that a data subject may file in the context of remediability, we define the trigger `TriggerOnComplaint` that triggers a set of specific actions to be undertaken by an auditor or/and a data controller.

6 VALIDATION

In this section we validate the accountability policy representation framework by extracting obligations from one of the use cases documented in the A4Cloud public deliverable DB3.1 (Bernsmed et al., 2013) and illustrate their representation in AAL and A-PPL.

6.1 The Health Care Use Case

This use case concerns the flow of health care information generated by medical sensors in the cloud. The system, which is illustrated in Figure 4, is used to support diagnosis of patients by the collection and processing of data from wearable sensors. Here, we investigate the case where medical data from the sensors will be exchanged between patients, their families and friends, the hospital, as well as between the different Cloud providers involved in the final service delivery.

In this use case the patients are the data subjects from whom (sensitive) personal data is collected. The hospital is ultimately responsible for the health care services and will hence act as one of the data controllers for the personal data that will be collected. The patients' relatives may also upload personal data about the patients and can therefore be seen as data controllers (as well as data subjects, when personal data about their usage of the system is collected from them). As can be seen in Figure 4, the use case will involve cloud services for sensor data collection and processing (denoted cloud provider "X"), cloud services for data storage (denoted cloud provider "Y") and cloud services for information sharing (denoted cloud provider "M"), which will be operated by a collaboration of different providers. Since the primary service provider M, with whom the users will interface, employs two sub-providers, a chain of service delivery will be created. In this particular case, the M platform provider will be the primary service provider and will act as a data processor with respect to the personal data collected from the patients. Also the sub-providers, X and Y, will act as data processors. The details of the use case is further described in DB3.1 (Bernsmed et al., 2013).

6.2 Obligations for the Use Case

We have identified a number of obligations for this use case, which needs to be handled by the accountability policy framework. Here we list three examples and we explain how they will be expressed in AAL and mapped into A-PPL. Note that the complete list of obligations is much longer, but we have chosen to outline those that illustrate the most important relationships between the involved actors. Due to space limitations we do not include the complete A-PPL policies here; the reader is referred to the project documentation (Garaga et al., 2013) to see the full policy expressions.

Obligation 1: The Data Subject's Right to Access, Correct and Delete Personal Data. According the Data Protection Directive (Directive, E. U., 1995), data subjects have (among others) the right to access, correct and delete personal data that have been collected about them. In this use case it means that the hospital must allow read and write grant access to patients as well as relatives with regard to their personal data that have been collected and stored in the cloud. There must be also means to enforce the deletion of such data.

The AAL expression catching the obligations associated to the patient is:

```
(MAY Patient.READ(D:Data)  OR
 MAY Patient.WRITE(D:Data) OR
 MAY Patient.DELETE(D:Data))
AUDITING
  MUST Auditor.AUDIT(hospital.logs)
IF_VIOLATED_THEN
  MUST Auditor.SANCTION(hospital)
```

The policy includes the `ALWAYS` operator since this property is expected to be true at any instant. The policy also includes the condition `D.subject=Patient` to express that this expression only concerns the personal data of the Patient. This policy also expresses the audit and rectification obligations that have to be ensured by an external Auditor.

Using the accountability policy representation framework, the AAL expression will be mapped into two different A-PPL expressions; one for permitting read and write access to the patients and another one for enforcing the data controller to delete the personal data whenever requested. Read and write access control is achieved through XACML rules. Regarding deletion of data, a patient can express data handling preferences that specify the obligation that the data controller has to enforce to delete the personal data. This obligation can be expressed using the A-PPL obligation action `ActionDeletePersonalData`, which will be used by the patient to delete personal data that has been collected about him.

An explicit audit clause implies that information related to the usage control property are logged (the amount and the nature of this information is not discussed here). Thus the audit clause is translated into an `AuditAction` which is responsible to manage the interaction with the auditor. This runs an exchange protocol with the auditor which ends with two responses: either no violation of the usage has been detected or a violation exists. In the latter, some rectification clauses should be specified.

In the sequel we only consider usage control clauses since the translation process for audit and rectification is similar to the previous example.

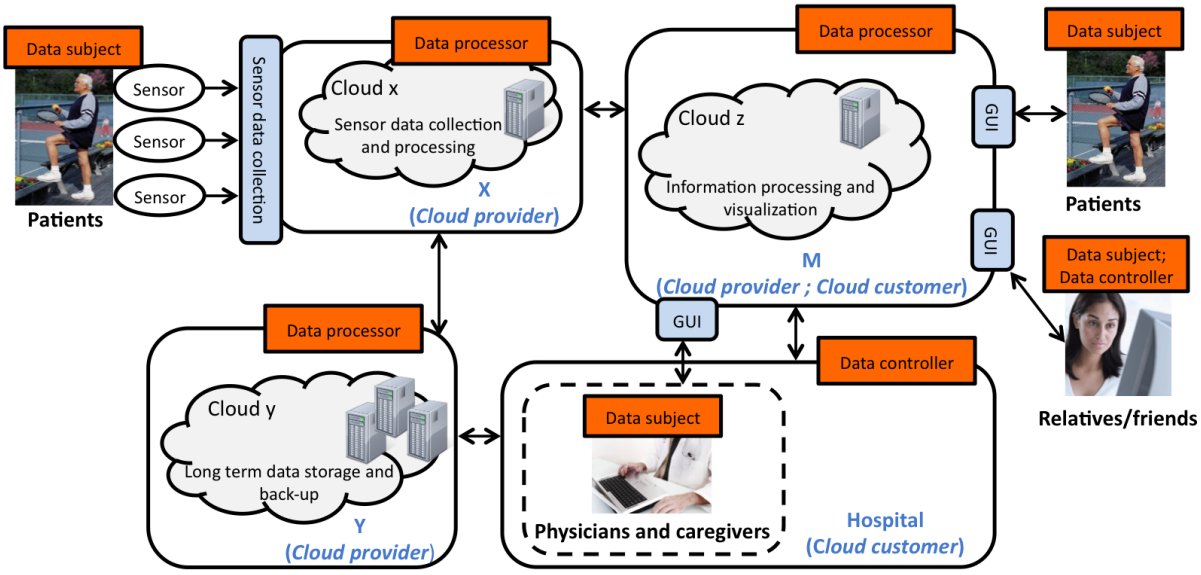

Figure 4: An overview over the main actors involved in the health care use case.

Obligation 2: The Data Controller Must Notify the Data Subjects of Security or Personal Data Breaches. This obligation defines what will happen in case of a security or privacy incident. In AAL it will be expressed by the hospital as:

```
MAY hospital.VIOLATEPOLICY() THEN
 (MUST hospital.NOTIFY[Patient]("incident") AND
  (ANY:Agent in hospital.relatives[Patient] THEN
    MUST hospital.NOTIFY[ANY]("incident")))
```

In A-PPL, such notification is expressed through the obligation action `ActionNotify`. It takes as parameters, the recipient of the notification (here, the data subject) and the type of the notification (here, security breach).

Obligation 3: The Data Processor Must, upon Request, Provide Evidence to the Data Controller on the Correct and Timely Deletion of Personal Data. To express the timely deletion of personal data, which in addition will be logged to be used as evidence, the following AAL expression can be used by the provider M:

```
MUST M.DELETE(D:Data) THEN
MUST M.LOGS("deleted", D, currentDate)
```

In A-PPL, the obligation trigger `TriggerPersonalDataDeleted` combined with the obligation action `ActionNotify` will notify the data subject of the deletion of its data. In addition, if necessary, the obligation action `ActionLog` will allow the provider M to log when personal data have been deleted.

The three examples that we have provided in this section represent a snapshot of the full power of AAL and A-PPL. In (Bernsmed et al., 2013) we outline more examples of obligations for the health care use case, which among other things demonstrate how informed consent can be gathered from the patients before their data is being processed, how the purpose of personal data collection can be specified and controlled, how the data processor M can inform the hospital of the use of sub-processors and how the data processors can facilitate for regulators to review evidence on their data processing practices.

7 CONCLUSION

Dealing with personal data in the cloud raises several accountability and privacy issues that must be considered to promote the safety usage of cloud services. In this paper we tackle the issue related to accountability obligations and preferences representation. We propose a cloud accountability policy representation framework. This framework enables accountability policy expression in a human readable fashion using our abstract accountability language (AAL). Also, it offers the means for their mapping to concrete enforcement policies written using our accountability policy language (A-PPL). Our framework applies the separation of concerns principle by separating the abstract language from the concrete one. This choice makes both contributions, i.e. AAL and A-PPL, self-contained and allows their independent use. The abil-

ity of our framework to represent accountability obligations was validated through a realistic health care use case.

Our future research work will focus on the mapping from AAL to A-PPL. As part of our implementation perspectives, we are currently working on two prototypes. An AAL editor that assists end-users in writing their preferences/obligations and implements the required artifacts to map them to concrete policies in A-PPL. We also started the development of an A-PPL policy execution engine that will be in charge of interpreting and matching A-PPL policies and preferences

ACKNOWLEDGEMENTS

This work was funded by the EU's 7th framework A4Cloud project.

REFERENCES

Aktug, I. and Naliuka, K. (2008). ConSpec – a formal language for policy specification. *In Electronic Notes in Theoretical Computer Science*, volume 197, pages 45–58.

Allam, D., Douence, R., Grall, H., Royer, J.-C., and Südholt, M. (2012). Well-Typed Services Cannot Go Wrong. *Rapport de recherche RR-7899, INRIA*.

Ardagna, C. A., Bussard, L., De Capitani Di Vimercati, S., Neven, G., Paraboschi, S., Pedrini, E., Preiss, S., Raggett, D., Samarati, P., Trabelsi, S., and Verdicchio, M. (2009). Primelife policy language. http://www.w3.org/2009/policy-ws/papers/Trabelisi.pdf.

Becker, M. Y., Malkis, A., and Bussard, L. (2010). S4P: A generic language for specifying privacy preferences and policies. *Microsoft Research*.

Bernsmed, K., Felici, M., Oliveira, A. S. D., Sendor, J., Moe, N. B., Rübsamen, T., Tountopoulos, V., and Hasnain, B. (2013). Use case descriptions. *Deliverable, Cloud Accountability (A4Cloud) Project*.

Bradner, S. (1997). IETF RFC 2119: Key words for use in RFCs to Indicate Requirement Levels. *Technical report*.

Breaux, T. D. and Anton, A. I. (2005). Deriving semantic models from privacy policies. In *Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY '05)*, pages 67–76.

Cranen, S., Groote, J. F., Keiren, J. J. A., Stappers, F. P. M., de Vink, E. P., Wesselink, W., and Willemse, T. A. C. (2013). An overview of the mCRL2 toolset and its recent advances. *TACAS'13*, pages 199–213, Berlin, Heidelberg. Springer-Verlag.

DeYoung, H., Garg, D., Jia, L., Kaynar, D., and Datta, A. (2010). Experiences in the logical specification of the HIPAA and GLBA privacy laws. In *9th Annual ACM Workshop on Privacy in the Electronic Society (WPES '10)*, pages 73–82.

Directive, E. U. (1995). Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data . http://ec.europa.eu/justice/policies/privacy/docs/95-46-ce/dir1995-46_part1_en.pdf.

Feigenbaum, J., Jaggard, A. D., Wright, R. N., and Xiao, H. (2012). Systematizing "accountability" in computer science. *Technical Report YALEU/DCS/TR-1452*, University of Yale.

Garaga, A., de Oliveira, A. S., Sendor, J., Azraoui, M., Elkhiyaoui, K., Molva, R., Önen, M., Cherrueau, R.-A., Douence, R., Grall, H., Royer, J.-C., Sellami, M., Südholt, M., and Bernsmed, K. (2013). Policy Representation Framework. Technical Report D:C-4.1, Accountability for Cloud and Future Internet Services - A4Cloud Project.

Haeberlen, A., Aditya, P., Rodrigues, R., and Druschel, P. (2010). Accountable virtual machines. In OSDI, pages 119–134.

Jagadeesan, R., Jeffrey, A., Pitcher, C., and Riely, J. (2009). Towards a theory of accountability and audit. In *Proceedings of the 14th European conference on Research in computer security*, ESORICS'09, pages 152–167, Berlin, Heidelberg. Springer-Verlag.

Kerrigan, S. and Law, K. H. (2003). Logic-based regulation compliance-assistance. In *International Conference on Artificial Intelligence and Law*, pages 126–135.

Knuth, D. E. (1964). backus normal form vs. backus naur form. Commun. *ACM*, 7(12):735–736.

Lamanna, D. D., Skene, J., and Emmerich, W. (2003). SLAng: A Language for Defining Service Level Agreements. In *Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems*, pages 100–, Washington, DC, USA. IEEE Computer Society.

Legislative Assembly of Ontario (1988). Freedom of information and protection of privacy act (r.s.o. 1990, c. f.31).

Marchiori, M. (2002). The platform for privacy preferences 1.0 (P3P1.0) specification. W3C recommendation, W3C. http://www.w3.org/TR/2002/REC-P3P-20020416/.

Métayer, D. L. (2009). A formal privacy management framework. *Formal Aspects in Security and Trust*, pages 1–15.

OASIS Standard (2013). eXtensible Access Control Markup Language (XACML) Version 3.0. 22 January 2013. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html.

Pearson, S., Tountopoulos, V., Catteddu, D., Südholt, M., Molva, R., Reich, C., Fischer-Hübner, S., Millard, C., Lotz, V., Jaatun, M. G., Leenes, R., Rong, C., and Lopez, J. (2012). *Accountability for cloud and other future internet services*. In CloudCom, pages 629–632. IEEE.

Pearson, S. and Wainwright, N. (2013). An interdisciplinary approach to accountability for future internet service provision. *International Journal of Trust Management in Computing and Communications*, 1(1):52–72.

Schneider, F. B. (2000). Enforceable security policies. ACM Transactions on Information and System Security, 3(1):30–50.

Sundareswaran, S., Squicciarini, A., and Lin, D. (2012). Ensuring distributed accountability for data sharing in the cloud. *Dependable and Secure Computing, IEEE Transactions on*, 9(4):556–568.

US Congress (1999). Gramm-leach-bliley act, financial privacy rule. 15 usc 6801-6809. http://www.law.cornell.edu/uscode/usc_sup_01_15_10_94_20_I.html.

US Congress (2002). Health insurance portability and accountability act of 1996, privacy rule. 45 cfr 164. http://www.access.gpo. gov/-nara/cfr/waisidx_07/45cfr164_07.html.

Wei, W., Du, J., Yu, T., and Gu, X. (2009). Securemr: A service integrity assurance framework for mapreduce. In *Proceedings of the 2009 Annual Computer Security Applications Conference*, pages 73–82, Washington, DC, USA. IEEE Computer Society.

Weitzner, D. J., Abelson, H., Berners-Lee, T., Feigenbaum, J., Hendler, J., and Sussman, G. J. (2008). Information accountability. Commun. *ACM*, 51(6):82–87.

Zhifeng Xiao, Nandhakumar Kathiresshan, Y. X. (2012). A survey of accountability in computer networks and distributed systems. Security and Communication Networks, 5(10):1083–1085.

Zou, J., Wang, Y., and Lin, K.-J. (2010). A formal service contract model for accountable SaaS and cloud services. In *International Conference on Services Computing*, pages 73–80. IEEE.