A Simulation Framework to Model Accountability Controls for Cloud Computing

Nick Papanikolaou
Security and Cloud Lab
HP Labs
Bristol, United Kingdom
Email: nick@nick-p.info

Thomas Rübsamen, Christoph Reich
Cloud Research Lab
Hochschule Furtwangen University
Furtwangen, Germany
Email: {ruet, rch}@hs-furtwangen.de

***Abstract*—In this paper, we present an implemented system to model and visually represent the functioning of accountability mechanisms for cloud computing (such as policy enforcement, monitoring, intrusion detection, logging, redress and remediation mechanisms) over provider boundaries along the supply chain of service providers. Service providers can use these mechanisms, among others, in a variety of combinations to address data protection problems in the cloud, such as compliance failures, losses of governance, lock-in hazards, isolation failures, and incomplete data deletion. The focus here is on technical mechanisms for the purposes of simulation (the currently implemented tool demonstrates policy enforcement, monitoring and logging); in general, an accountability approach requires a combination of technical measures and legal and regulatory support, of course. We survey existing work on accountability in the cloud and discuss ongoing research in the context of the Cloud Accountability project. We discuss modelling considerations that apply in this context namely, how accountability may be modelled statically and dynamically. Details of the current implementation of the Accountability Simulation Engine (ASE), and the first version of a graphical animation of data flows in the cloud, are described.**

Keywords–accountability; data protection; modelling language; simulation; visualisation; sticky policies; policy enforcement; logging; redress

I. INTRODUCTION

In this paper, we present the background and modelling considerations associated with Accountability Simulation Engine (ASE), a simulation framework to model and visualize accountability mechanisms for cloud computing. We will discuss the motivation and objectives behind ASE, as well as the features that have been implemented so far. As this is still ongoing work, the primary purpose of the paper is to inform the community and to impart some structure on the development activities; a detailed discussion of future work has also been included.

The starting point for this work is the realization that both cloud computing service providers, as well as customers of cloud computing services, need to have a good understanding of the controls that may be used for managing data flows in the cloud while complying with prevailing data protection laws, rules and regulations, as well as industry standards, best practices, and corporate data handling guidelines in an efficient yet demonstrable manner. The massive scale of cloud computing infrastructures, as well as the enormous complexity of legal and regulatory compliance across multiple jurisdictions, makes this a significant and difficult challenge that service providers, customers, regulators and auditors need to meet on a continual basis.

Accountability for cloud computing service provision is emerging as a holistic approach to this set of issues, and is being actively developed in the context of the Cloud Accountability Project (A4Cloud) [1]. Drawing on a multitude of sources, including legal and regulatory frameworks for accountability, as well as technical solutions for achieving data protection compliance, this project aims to provide cloud service providers, auditors, regulators and others with a concrete set of tools for achieving accountability. The simulation framework described in this paper is a research tool whose purpose is to demonstrate how such tools might work, and what problems they are intended to address.

As seen in Section II, accountability encompasses a number of different controls that may be used by a cloud service provider to ensure appropriate governance of their customers data. By developing a simulation of how these controls might or should function, we can reason about their necessity and suitability to particular data handling scenarios. We are mainly interested in technical, automated means of achieving accountability; higher-level mechanisms (e.g., legal rulings or precedents, new regulations, ethical guidelines) are only implicitly modelled as rules incorporated in technical enforcement mechanisms (such as privacy or access control policies).

In order to better understand what form of simulation would be appropriate, we surveyed a number of existing simulation tools and frameworks (Section III). We identified two classes of tools discrete-event modelling formalisms with mostly textual output, as well as visual simulation tools, which permit rapid prototyping, and the creation of graphical animations.

An important part of this work has been identifying what components a suitable simulation model might include, as well as what use cases and scenarios might best illustrate the functionality of accountability mechanisms. These topics are discussed in Section IV. It is interesting to note that there are both static and dynamic aspects of accountability, and different types of simulation are suited to these aspects.

The next section details our model namely, what actors, behaviours and relationships we have chosen to include. The design choices are not definitive, and are likely to vary across use cases. However, this section establishes which kinds of issue could be demonstrated during a simulation, and which responses or mechanisms are appropriate when an accountability-based approach is taken. Section V describes our current implementation of an accountability simulation engine (ASE), which comprises (i) a domain-specific mod-

Copyright (c) IARIA, 2014. ISBN: 978-1-61208-338-4

elling language for accountability scenarios, (ii) an actual simulator for accountability related events, (iii) a web-based user interface for inputting scenario descriptions and observing simulation output, and (iv) a web service which links (ii) and (iii) together. An example of ASEs functionality is given in Section VI, with a simulation of the dynamics of a simple cloud service provision chain. Finally, we conclude with a discussion of future work; this includes prototyping a visual animation of data flows using existing simulation tools and extending the current implementation of ASE with graphical output.

II. The Need for Accountability in the Cloud

As identified by Pearson [2], accountability "for complying with measures that give effect to practices articulated in given guidelines" has been present in many legal and regulatory frameworks for privacy protection; certainly the notion originates from the data protection context, and carries with it the idea of responsible data stewardship. The Galway project [3] attempted to define accountability in this context as follows:

Accountability is the obligation to act as a responsible steward of the personal information of others, to take responsibility for the protection and appropriate use of that information beyond legal requirements, and to be accountable for any misuse of that information.

Pearson [2] observes that the key elements of notion of accountability implied by this definition are *transparency, responsibility, assurance and remediation*. In Pearson and Wainwright [4] it is argued that, to support these elements, it is possible to co-design legal and technical controls for cloud service providers belonging to three categories (i) preventive controls (e.g., risk analysis decision support tools, policy enforcement using machine-readable policies, privacy enhanced access control and obligations), (ii) detective controls (e.g., intrusion detection systems, policy-aware transaction logs, and reasoning tools, notification mechanisms), and (iii) corrective controls (e.g., liability attribution tools, incident management tools). Other categories of controls exist for different kinds of participants in a cloud service provision ecosystem, including end users and regulators.

Our interest is in creating a simulation framework, which enables us to address concerns such as the following:

- What problems can arise in a cloud service provision chain when controls such as those described above are absent from service providers infrastructures;
- What benefits the adoption of such controls can have on service providers operational responses to problems, such as data breaches;
- How accountability can be maintained along a supply chain of cloud service providers;
- What potential impact the introduction of a new control can have on a service providers operations.

While the goal of designing the simulation is to demonstrate the added benefits of adopting an accountability approach, in order to do so it is necessary first to identify the problems and events of interest that this approach provides responses for; both the events and the responses to these events can then be explicitly accounted for in the simulation model. Additionally, audit plans can be derived from this model more precisely and more fully.

A. Data Protection Problems

Based on the risk categorization presented by Haeberlen et al. in [5], we identify here five typical classes of data protection problems that cloud service providers need to mitigate:

- Compliance failures
- Losses of governance (e.g., data breaches)
- Lock-in hazards
- Isolation failures
- Incomplete data deletion

Compliance failures. As mentioned in the introduction, cloud service providers need to ensure compliance with prevailing laws and regulations [6][7] in the jurisdictions where customers data are stored. This is a non-trivial matter, given that cloud data centres are located in multiple, different locations across the globe, and data often needs to be relocated from one data centre to another for efficiency, bandwidth or other considerations. To ensure compliance on an ongoing basis, applicable local rules need to be checked before, during and after data relocation and evidence has to be given by audits. What makes this particularly complex is that rules are not consistent everywhere, and often transformations need to be applied to the data itself (e.g., in the case of anonymisation of personal data) before a transfer can occur. Any failure to comply with laws and regulations carries significant consequences for the reputation and profits of a service provider; therefore, it is of paramount importance to ensure immediate corrective action if any case of non-compliance is detected (e.g., by audits).

Compliance hazards are not confined to legal and regulatory requirements, of course; in order to maintain industrial certifications and badges, service providers need to ensure compliance with appropriate industry standards, whether specific to cloud computing practices, data handling practices, or quality control, among other things. These are typical tasks of a cloud audit system. Failure to maintain such compliance can result in loss of accreditation and, again, loss of reputation for a cloud service provider.

Losses of governance. As data flows from service provider to service provider and beyond, problems can occur at the boundaries: the controls employed by a service provider can only directly ensure appropriate governance of data within the boundaries of that providers infrastructure. The primary cloud service provider within a service provision chain namely, the main cloud service provider in a chain, interacting directly with an enterprise customer loses control over data as it is handed over to that customer. If an entity with malicious intent gains control at the cloud service provider customer interface, this loss of governance on the part of the cloud service provider can have serious consequences for the confidentiality, integrity and availability of the customers data. Data provenance mechanisms, which are not restricted to a single cloud service provider might help to mitigate these problems [8].

Lock-in Hazards. Cloud service providers can create vendor lock-in issues for customers by forcing them to use particular formats for data. If those formats are not widely accepted, it may be very difficult to extract and convert the data for use further down the cloud service provision chain. A hazard can occur during an attempted conversion of data to another format particularly if the format in which the data is stored is encrypted, as such encryption is necessarily lost during the process, thus revealing the data to a potential attacker.

Isolation failures. In a multi-tenanted cloud environment, multiple customers data are stored on the same infrastructure by a cloud service provider; a standard contractual requirement in such a scenario is that isolation of different customers data and operations is maintained; in the absence of such isolation, attacks and hazards affecting one customer can affect another, due to interactions occurring on the common underlying infrastructure. Isolation failures can cause rapid propagation of viruses, worms and similar infections, affecting multiple customers data and damaging the cloud service providers reputation.

Incomplete data deletions. Data retention laws, typically, require cloud service providers to maintain customer data for a certain period of time after service has terminated. After this period has lapsed, the data has to be deleted from the cloud service providers infrastructure and, depending on the contractual terms applicable for the particular customer, disposed of using particular technical means. Failure to delete data in accordance with the relevant contractual terms can have serious consequences, and could even cause integrity issues for new customers using the same infrastructure if only partially overwritten.

Data protection problems such as the above are illustrative of issues that we need to instantiate in a simulation framework for accountability in the cloud.

B. Addressing Data Protection Problems: Controls for Accountability

While an accountability-based approach to data governance combines a number of mechanisms, ranging from high-level, legal obligations, all the way down to technical controls, our interest is in demonstrating just the latter namely, how technical measures, particularly automated tools, can be introduced into a cloud service providers infrastructure to address issues such as those presented in the previous section. As we have seen, we can classify controls into three categories, namely, preventive, detective, corrective depending on whether they are intended as measures to be deployed prior to or after a problem occurs.

Next, we describe the types of controls that we are modelling in the ASE framework.

Among preventive controls, we focus on **policy enforcement mechanisms**, in particular tools that allow organisations to ensure that pre-defined, machine-readable policies are enforced automatically within their information technology (IT) infrastructures. For the purposes of simulation, we will define **accountability policies** and the types of rules that may be encountered in such policies.

Detective controls usually take the form of background processes or aspects in a system; such controls can be active or passive, or some combination of the two. Active controls such as **monitoring** or **intrusion detection** react to particular events and patterns of behaviour, such as threats or data breaches. Passive controls include, for example, **logging tools**, whose function is to record all events that take place (with the source of the event, type of event, and other details) so that a service provider can (i) trace particular activities and identify sources of problems (this relates to attribution capabilities needed for accountability), (ii) prove compliance (with rules, regulations, standards, best practices and more) to external parties such as auditors. An example for such passive controls is Amazon's AWS CloudTrail [9], which provides cloud customers with an API call history and logs.

For corrective controls, there is a lack of previous work; mechanisms that are relevant are tools for providing **redress** to customers in cases where data protection problems have not been mitigated by preventive or detective controls. **Incident management tools** are relevant here, but exactly what remedies or responses are appropriate for different types of incidents remains an ongoing research challenge. For the purposes of simulation using ASE, we will assume that financial remedies (including **payment of fines** and other **penalties** for service providers) are suitable responses. The introduction of additional preventive measures, such as storage encryption depending on the type and sensitivity of stored data may also be a response. However, for the purpose of this paper, this is out of scope.

III. Review of Selected Current Simulation Tools and Platforms

Although we have developed the ASE simulation framework from the ground up, we have surveyed and experimented with a number of existing simulation tools; only discrete-event simulation tools have been considered, since our interest is in understanding behaviours and mechanisms that can be effectively modelled using this paradigm.

The tools of interest include software libraries providing dedicated simulation functionality, such as built-in data structures for event queues, random number generation using different probability distributions, timing information and more, as well as visual tools for designing simulations using predefined components.

Discrete-event simulation is detailed in the authoritative text by Law [10], which also includes a library for use in C programs, named *simlib*. This enables one to make use of commonly used data structures for simulation, as mentioned above. There are other libraries with similar capabilities, and indeed *simlib* has been rewritten and adapted for use in other programming languages (e.g., Brian J. Huffman has produced a Java version of the library [11]). We are also aware of the Java-based *Greenfoot* framework [12], which allows simulations to be prototyped easily; so far, we have not found a way to turn *Greenfoot* code into web-based applications, which was desirable for our purposes.

An interesting, more recent Java-based simulation library that we experimented with is the agent-based simulation framework MASON [13], which also includes graphical animation

capabilities. The distinctive feature of MASON is that it allows one to produce animated graphical user interfaces to demonstrate interactions in multi-agent systems. Since the demonstrations we have been building are currently relatively small-scale, as opposed to its usual applications, we abandoned MASON early on. Nevertheless, its modular design and graphical capabilities may well be used in future versions of the accountability simulator.

Another approach that we considered included the use of industrial-strength visual simulation tools, such as MATLAB™-Simulink™ [14] and Simio™ [15]. Using the trial version of Simio™, we were able to produce a simple, 3D graphical animation of data flows between cloud service providers, as shown in Figure 1. We were not able to simulate accountability mechanisms using the trial version, as this would require building/coding a significant number of Simio™ processes, a feature that is limited. This will be included in our future work. However, we were able to gain visual insight into the nature and purpose of the simulation, which will be discussed in the next section.

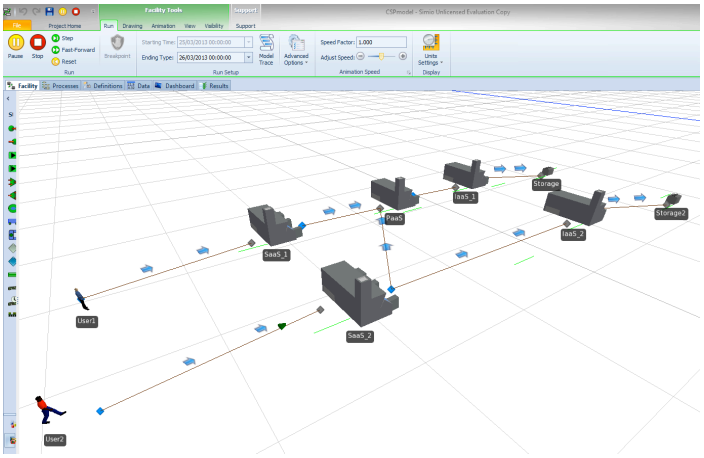

Figure 1. Screenshot of simulated data flows between cloud service providers

IV. Modelling Considerations

As we have seen in previous sections, in order to build a simulation of accountability in the cloud, we need to identify a way to show (i) data protection problems that arise in cloud ecosystems, and (ii) how accountability controls or mechanisms work to mitigate and respond to these problems. The objective of this work is to build a graphical simulation which can provide insight for a variety of stakeholders, including cloud service providers, regulators, auditors and even the general public interested in how accountability can be achieved in a complex chain of cloud service provision. But what should be the underlying conceptual model of the simulation? There are different aspects to consider here.

A. Static Modelling: Actors and Relationships

One aspect to consider is the set of relationships (and the properties of these relationships) between different cloud service providers in a service provision chain. From the data protection point of view, there are different roles for cloud service providers when it comes to handling personal data terms used in the European Data Protection Directive 95/46/EC [7] for these roles are *data controller* and *data processor*. Depending on the service offering, providers may take one or both of these roles, with complex and ambiguous cases arising frequently. Modelling what this implies in terms of concrete obligations for cloud service providers is what we will refer to as static *modelling of accountability*.

TABLE I. The possible roles that the different kinds of actors can take on in a given scenario as per our model.

Actor Type	Possible Roles
Individual	Data subject
Cloud service provider	Data controller Data processor
Third party	Data controller Data processor Accountability Agent
Auditor	(Auditor) Accountability Agent
Regulator	(Regulator) Accountability Agent

The static modelling of a cloud service provision chain involves identified actors, roles and responsibilities and the relationships between them.

1) Actors, Roles and Responsibilities: In our model, in a cloud ecosystem there are five different kinds of actors **individuals, cloud service providers, third parties, auditors** and **regulators**. We classify the different types of roles that these actors may take on in a particular scenario into six kinds **data subject, data controller, data processor, accountability agent, auditor** and **regulator**. A particular scenario is defined as a specified set of roles for a specified set of actors.

The possible roles that the different kinds of actors can take on in a given scenario as per our model are defined in Table I.

The roles that we have included take into account the static modelling discussion in Section IV-A. *Accountability agent* represents a role that is intended to encompass internal oversight activities within an organization (e.g., self-auditing), as opposed to the roles of *auditor* (an external entity performing an audit on behalf of enterprise) and *regulator* (typically a government entity responsible for setting, implementing and monitoring standards), which by definition correspond to oversight external to an organization; note that we model two classes of organizations here cloud service providers and third parties, the latter being providers of non-cloud services. The distinction becomes clearer when we consider relationships that can exist between actors.

2) Relationships: Cloud service providers are characterized in the model by the kind of relationship they have with other providers, in particular, what kind of service they offer to others. A cloud service provider provides one of three kinds of service: **IaaS** (infrastructure-as-a-service), **PaaS** (platform-as-a-service), **SaaS** (software-as-a-service). These are the only kinds of relationships considered here between cloud service providers. Third parties are entities that enter into complex contractual relationships with cloud service providers, relationships that are not of the same kind. Further investigation is needed here; but, for the purposes of modelling and simulation we do not need to restrict the kinds of relationship that third parties may have (with each other and with cloud service

Copyright (c) IARIA, 2014. ISBN: 978-1-61208-338-4

providers).

B. Modelling System Dynamics: Data Transfers and Accountability Mechanisms

While static modelling would enable us to simulate what effect particular assumptions might have on the obligations of a cloud service provider, *modelling system dynamics* enables us to simulate data flows between cloud service providers, data protection problems and their consequences when accountability controls are in place (and similarly when such controls have not been introduced). For a dynamic simulation, the main entities that need to be modelled are *personal data*; at each step of the simulation, personal data flow through a chain/sequence of service providers, which are predefined, and together constitute a model of a real-world cloud service provision chain. The purpose of the simulation becomes to show what happens to the data as they flow through the chain, and what effect these flows have on properties of the overall system.

So, what is our model? The entities modelled have been discussed in the previous subsection, along with their relationships; next, we discuss their expected behaviours, and the types of issues or problems that can be simulated using our model, and the responses that different entities can have and should have to such problems if accountability is to be achieved.

1) Behaviours corresponding to different types of role: First, consider the behaviours of individual data subjects. In our model, a data subject is an entity that can engage in one of the following actions at any time during a simulation:

- Create data (a datum is modelled simply as a pair of strings an identifier and a value)
- Modify/edit data
- Delete data
- Change preferences regarding usage of data (initially, a data subjects policy is simply a statement of for what uses data can be processed, and whether the data can be shared with third parties)
- Request summary of data and preferences held

For service providers, which are typically data controllers or data processors, the following actions are possible:

- Store data
- Encrypt and store data
- Decrypt data
- Check preferences and share data
- Create new policy over data
- Generate log of activity over data
- Enable/disable logging mechanisms
- Enable/disable monitoring mechanisms
- Enable/disable policy enforcement mechanisms

Regulators and auditors are modelled as having the following possible actions:

- Check compliance of data controller/processor with a specified rule or set of rules
- Check compliance of organizational policies with minimum requirements
- Create new rules specifying allowed uses of customer data, and penalties/remedies in case of non-compliance or other problem
- Create new rules specifying mechanisms that must be used to protect data subjects, and penalties/remedies in case of non-compliance or other problem
- Enforce penalty or other remedy in case of non-compliance or other problem
- Audit a data controller/processors system logs (esp. check origin, route, destination of data; intended use; protection mechanisms used, whether customers preferences were enforced)

Accountability agents are initially to be modelled as a variant of auditor, with the only difference that they cannot perform enforcement, only (implicitly) inform the organization they are associated with of any events of interest (e.g., failures, non-compliance). Further work may reveal other actions/events of interest.

2) Simulated Issues: In the simulation, we should be able to represent and visualize some of the issues discussed previously in Section II.A. Compliance failures, data breaches and direct attacks on a service providers infrastructure are specific events that we have so far considered in this work.

3) Simulated Responses: In Table II, we can see how the different accountability mechanisms considered in our model can help to address the simulated issues. We note that this list is not exhaustive, as it only includes the mechanisms we have considered so far; other mechanisms could include, for example, automated tools for punishment or remediation; also, we have so far avoided detailing what types of rules are allowed in policies. In the Cloud Accountability Project, which is the context in which the simulation has been built, there is an ongoing work on developing an accountability policy language; for the purposes of our simulation, we have so far assumed that rules restrict to whom and under what conditions data can flow; the distinction between data controller and data processor may well imply additional restrictions, and similarly there are restrictions on when data can be transferred to third parties (this is modelled as a preference that data subjects can set).

As we can see from the table, when none of the accountability mechanisms are enabled in a simulation, none of the problems considered trigger any response (thus allowing hazards and failures to occur). Of interest is the fact that hazards can then propagate (cascade) from one provider to another, and/or to any third parties. All other cases cause a response, thus demonstrating how accountability mechanisms work in practice.

V. IMPLEMENTATION

We have implemented three software components as part of the accountability simulator: a simulation engine, a web-based animation of data flows between cloud service providers,

TABLE II. PROBLEM AND THE SPECIFIC RESPONSES TRIGGERED BY ACCOUNTABILITY MECHANISMS IN THE SIMULATION MODEL

Problem	Mechanism			
	None	Policy enforcement	Monitoring	Logging
Compliance failure	X	All problems correspond to specific policy violations; Policy violation will be detected; Parties notified	Patterns of non-compliance can be detected	All failures will be logged and shown to auditors
Data breach	X		Monitoring interaction of service providers with untrusted third parties can provide advance warnings	All breaches will be logged and shown to auditors
Attack	X		Intrusion detection systems can be used to thwart attacks	All attacks will be logged and shown to auditors

and a web service that draws data from the simulation engine. Currently, we are continuing implementation work until all three components have been fully integrated. In this section, we present the functional structure of the simulator and then detail each of the implemented components.

The input file is a description of a scenario to be simulated, and is written in the accountability model description language, described in the next section. A scenario consists of a specified set of actors (so far, we have not made the distinction between actors and roles in the language, but this is forthcoming in future versions), a specified set of relationships, configuration of options/parameters and the triggering of actions of particular actors.

When an input file is supplied to the simulator (via a web-based interface or through the command line), its contents are parsed using the language interpreter, which invokes appropriate methods in the accountability simulation engine. The accountability simulation engine contains the current state variables and the log of events executed so far; it constitutes the *backend* of the application and is written in plain Java.

In order to feed the state of the simulation, timings and outcomes to the web-based user interface, we have implemented a RESTful web service using the Java-based Restlet EE framework [16].

The initial version (designated *v1*) of the web-based user interface was implemented using HTML forms, and the data it receives from the web service consist of plain text strings that are displayed and updated as the simulation progresses, without any graphical animation.

The latest version (designated *v2*) of the web-based user interface has been developed separately, as a graphical animation, and work is ongoing to link it up to the web service. This will be discussed further in Section V-C below.

A. Accountability Model Description Language

Scenarios to be simulated are described by the user of the simulator in a custom modelling language we have built for this purpose. At this stage of development we have only included a core set of commands and mechanisms that can be included in scenarios, but we expect to add constructs in the language so as to allow inclusion of detailed privacy policies, access and usage controls, and other features. In particular, in the Cloud Accountability Project there is ongoing work on developing a dedicated accountability policy language, and it is likely that the constructs of that language will be incorporated into the language of the simulator.

Listing 1 shows the productions for the languages grammar, using the syntax of the SableCC [17] parser generator that we have been using to build the interpreter. The nonterminals in the grammar are *command* (for top-level commands), *declare* (used to declare actors of different types), *type* (representing the different actor types, namely user, cloud service provider, auditor and regulator), *servicetype* (for the different types of cloud service), *objectaction* (this is for expressions representing a property or action of a given actor), action (properties or actions), mode (for data protection problems that can be simulated using the trigger command), *mech* (for accountability mechanisms that can be enabled or disabled as needed). Commands *setgraph*, *setpolicy* and *setconstraint* are experimental; the command *graph* allows us to access the internal data structure of the accountability simulation engine and visualize it using AT&T GraphViz [18].

B. Accountability Simulator (Backend component)

The accountability simulation engine is responsible for maintaining and updating the current state of the simulation, and currently its main visible function is to display that state on the console or supply it (through a web service) to another application.

We can denote the internal state of the simulation engine by a tuple (see Equation 1)

$$(A, T, \rho, \sigma, M, \xi) \tag{1}$$

where A is the set of actors that have been declared, T is the set denoting the types of the actors, ρ is the set of relationships between cloud service providers (the only relationships modelled are between these types of actor), σ is the store of data values held by the different actors (indexed by A), M is the set of accountability mechanisms enabled and ξ is the output stream (this represents, e.g., the standard output or a pipe to another application, or a web service).

```
command =
{ declaration } declare |
{ custdeclaration } customer lparen
        [ fst ]: identifier [q]:comma
        [snd]: identifier [z]:comma
        servicetype rparen |
{action} objectaction |
{ trig } trigger mode identifier |
{setgraph} setgraph arglist |
{setpol} setpolicy lparen identifier comma
        str rparen |
{setcons} setconstraint lparen [ fst ]: identifier
        [q]:comma [snd]: identifier [z]:comma
        str rparen |
{enablemech} enable mech |
{disablemech} disable mech |
```

```
{graphing} graph;

declare = { declplain } type identifier |
          { declqualified } type identifier str ;

type = {userdec} user |
       {cspdec} csp |
       {auddec} auditor |
       {regdec} regulator ;

servicetype = { iaastype } iaas |
              {paastype} paas |
              {saastype} saas ;

objectaction = { actionref } identifier dot action ;

action = { actionsenddata } senddata
         lparen identifier comma str rparen |
         { evalstate } state ;

mode = {databreach} databreach |
       {attack} attack ;

mech = {polenfmech} polenf |
       {logmech} logging |
       {monmech} monitoring;
```

Listing 1. SableCC grammar of the Accountability Simulator input language (only the main productions are shown)

The output of the simulation depends on which accountability mechanisms have been enabled; if no mechanisms are enabled (in which case the value of M above would be the empty set, $\emptyset$), then there is no change in the output ξ when a problem is triggered. However, when mechanisms are enabled and a problem is triggered, the effect (as described in Table II) is made visible on the output. In other words, the Function of the simulator (see Function 2) can be summarized operationally as a state transition of the form:

$$(A, T, \rho, \sigma, M, \xi) \rightarrow (A, T, \rho, \sigma', M, \xi') \qquad (2)$$

such that ξ' differs from ξ as it contains a notification of a compliance failure, data breach or attack when the trigger command is issued and one of the following is true:

$$(\{policyenforcement : enabled\} \in M) \; or$$
$$(\{logging : enabled\} \in M) \; or$$
$$(\{monitoring : enabled\} \in M)$$

It would not be difficult to use the above notation to derive a full operational semantics for the simulator, but for our purposes here it is sufficient to note that the main function of the tool, which is to behave differently depending on which type of problem is being simulated, and which mechanisms are enabled. So far we have assumed ξ represents textual output, namely strings describing the overall system state, such as lists of actors, their data values and more. Of course, the exact output consists of messages corresponding to the responses shown in Table II. In the next section we turn to work we have done on developing a graphical visualization.

C. Web-based Front-End

The vision for the web-based user interface has always been to have a graphical animation of data flows between individuals, cloud service providers, auditors and regulators and third parties. Demonstrating flows of data and the changes that occur to data and providers in the process emphasizes the dynamic aspect of the simulation. A screenshot of our current prototype of version v2 is shown in Figure 2.

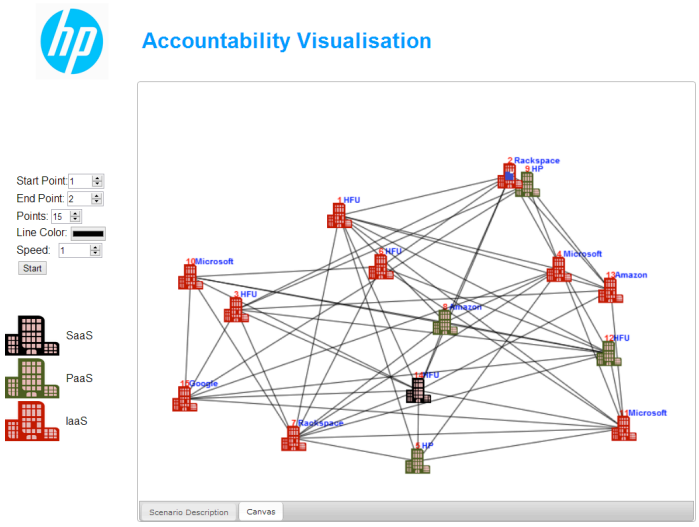

Figure 2. Visualisation front-end for the Accountability Simulator

It is very important to note that this version has been developed as a separate, standalone animation. Thus, it has not yet been linked to the web service component and, hence, the main simulation engine. However, it does show an instantiation of a random set of cloud service providers of different kinds (IaaS, PaaS, SaaS, and how a data item can be routed between providers. In the animation, the scenario is assumed to be random, rather than specified in the accountability modelling language; this is changing presently.

We expect to have dedicated controls (form buttons) to trigger particular data protection problems, and panels showing the responses produced by the simulator. In the screenshot in Figure 2 two tabs are shown at the bottom. While the (currently random) animation is shown on the *Canvas* tab, the other tab (titled *Scenario Description*) will allow the user of the simulation to supply an input file in future, written in the accountability modelling language of Section V-A, and this will be used to generate the animation in the next version.

VI. An Example Input File

Listing 2 shows an example input file that we have tested with the current version of the simulator. It describes a scenario in which there are two individual users, four cloud service providers, an auditor and a regulator. The relationships between the individuals and service providers are then declared. John and Mary then create some data, which is sent and stored on the specified service providers infrastructure. In line 15 a data breach problem is triggered; this has no effect when simulated as no accountability mechanisms have been enabled; the remaining lines enable different mechanisms, and trigger

a data breach and attack. Naturally, the simulator produces a sequence of long warnings when interpreting lines 17, 20 and 22, as the policy enforcement, monitoring and logging mechanisms kick in.

```
User john "John Wayne";
User mary "Mary Wollstonecraft";
CSP salesforce " Salesforce .com";
CSP amazon "Amazon Web Services";
CSP rackspace "Rackspace";
CSP hpcs "HP Cloud Services";
Auditor kpmg "KPMG";
Regulator cnil "CNIL";
Customer(john,amazon,SaaS);
Customer(mary, salesforce , SaaS);
Customer(rackspace,hpcs,IaaS);
Customer( salesforce , rackspace , PaaS);
john.Senddata(amazon,"somedata");
mary.Senddata(hpcs,"marydata");
Trigger Databreach salesforce ;
Enable Polenf;
Trigger Databreach salesforce ;
Enable Monitoring;
Enable Logging;
Trigger Databreach salesforce ;
amazon.State;
Trigger Attack amazon;
```

Listing 2. Example script written in the accountability simulation language.)

VII. Conclusion and Future Work

In this paper, we have presented the design and implementation of a simulator for accountability mechanisms in the cloud. We have discussed data protection problems, and how mechanisms for accountability such as policy enforcement, monitoring and logging can help to address such problems; the simulator we have built is a tool to assist understanding and modelling of real-world scenarios and will hopefully be a useful aid to cloud service providers, regulators and end users as it is extended with more features.

Future work will focus on integrating the v2 web-based UI with the accountability simulation engine and web service, and enriching that UI with more controls. Subsequently we will work on animating the accountability mechanisms and modelling additional ones.

It is also worth noting that a new EU Data Protection Regulation will eventually replace the current directive, which is under discussion in the European Parliament. This is likely to include new accountability rules and obligations, and must be taken into consideration in future work. For the purposes of this paper, however, we have focused on modelling the dynamics of accountability controls and how they impact data and data flows in cloud infrastructure.

Acknowledgment

Nick Papanikolaou wishes to cordially thank Siani Pearson and Nick Wainwright for their support and feedback during the development of this work. Special thanks are due to Fabian Reich, who spent a month at HP Labs working with Nick Papanikolaou and actually coded version v2 of the web-based GUI, navigating the vagaries of the HTML5 Canvas control. This work is supported by the European FP7 Programme A4CLOUD: Accountability for the Cloud and Other Future Internet Services.

References

[1] "Cloud accountability project (a4cloud)," http://www.a4cloud.eu/, [retrieved: 2014.04.08] 2014.

[2] S. Pearson, "Toward accountability in the cloud," Internet Computing, IEEE, vol. 15, no. 4, pp. 64–69, July 2011.

[3] Centre for Information Policy Leadership as Secretariat to the Galway Project, "Data protection accountability: The essential elements - a document for discussion," http://www.informationpolicycentre.com/files/Uploads/Documents/Centre/Centre_Accountability_Compendium.pdf, [retrieved: 2014.04.08] 2009.

[4] S. Pearson and N. Wainwright, "An interdisciplinary approach to accountability for future internet service provision," International Journal of Trust Management in Computing and Communications, vol. 1, no. 1, pp. 52–72, 01 2013.

[5] T. Haeberlen, L. Dupre, D. Catteddu, and G. Hogben, "Cloud computing: Benefits, risks and recommendations for information security," enisa - European Network and Information Security Agency, Tech. Rep., 2012.

[6] "Apec privacy framework," http://publications.apec.org/publication-detail.php?pub_id=390, [retrieved: 2014.04.08] 2005.

[7] "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://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:en:HTML, [retrieved: 2014.04.08] 1995.

[8] O. Q. Zhang, M. Kirchberg, R. K. L. Ko, and B. S. Lee, "How to track your data: The case for cloud computing provenance," HP Labs, Tech. Rep., 2012.

[9] "Amazon aws cloudtrail," https://aws.amazon.com/de/cloudtrail/, [retrieved: 2014.04.08].

[10] A. Law, Simulation Modeling and Analysis (McGraw-Hill Series in Industrial Engineering and Management). McGraw-Hill Science/Engineering/Math, 2006.

[11] B. J. Huffman, "An object-oriented version of simlib (a simple simulation package)," Informs Transactions on Education, vol. 2, no. 1, pp. 1–15, 2001.

[12] M. Koelling, Introduction to Programming with Greenfoot: Object-Oriented Programming in Java with Games and Simulations. Pearson Education, 2009.

[13] "Mason multiagent simulation toolkit," http://cs.gmu.edu/~eclab/projects/mason/, [retrieved: 2014.04.08].

[14] "Matlab and simulink," http://www.mathworks.co.uk/products/simulink/, [retrieved: 2014.04.08].

[15] "Simio," http://www.simio.com/, [retrieved: 2014.04.08].

[16] "Restlet framework," http://restlet.org, [retrieved: 2014.04.08].

[17] E. Gagnon, "Sablecc, an object-oriented compiler framework," http://www.sablecc.org/, [retrieved: 2014.04.08].

[18] "Graphviz – graph visualization software," http://www.graphviz.org, [retrieved: 2014.04.08].

Copyright (c) IARIA, 2014. ISBN: 978-1-61208-338-4