Quantitative Analysis of Opacity in Cloud Computing Systems

Wen Zeng and Maciej Koutny

Abstract—Federated cloud systems increase the reliability and reduce the cost of the computational support. The resulting combination of secure private clouds and less secure public clouds, together with the fact that resources need to be located within different clouds, strongly affects the information flow security of the entire system. In this paper, the clouds as well as entities of a federated cloud system are assigned security levels, and a probabilistic flow sensitive security model for a federated cloud system is proposed. Then the notion of opacity — a notion capturing the security of information flow — of a cloud computing systems is introduced, and different variants of quantitative analysis of opacity are presented. As a result, one can track the information flow in a cloud system, and analyze the impact of different resource allocation strategies by quantifying the corresponding opacity characteristics.

Index Terms—federated cloud computing, internet of things, opacity, security policy, information flow

1 Introduction

The extent and importance of cloud computing is rapidly increasing due to the ever increasing demand for internet services and communications. Instead of building individual information technology infrastructure to host databases or software, a third party can host these on its large server clouds. In addition, organizations may wish to keep sensitive information on their more restricted servers rather than on the public ones. This has led to the introduction of federated cloud computing in which both public and private cloud computing resources are used [1].

A federated cloud deploys and manages multiple cloud computing services, with various computational resources being allocated to different clouds for both security and business concerns. Although a federated cloud system (FCS) can increase the reliability and reduce the cost of computational support to an organization, the large number of services and data stored in the clouds creates security risks due to the dynamic movement of data, connected devices, and users between various cloud environments. As a result, it is necessary to track and control the information flow. In order to make such information and data traceable, one needs a formal model describing the information flow security within FCS. In this paper, we will introduce a probabilistic transition system representation of the information flow in FCS, and then investigate security properties of the information flow using *opacity*, which a notion capturing the security of information flow.

There has recently been a significant interest in building secure service-based systems and, on the other hand, in measuring the security of information flow in programming languages. The Bell-LaPadula model [2] is a state machine model which is used to enforce access control in government and military applications. In [3], the author applied this model to workflow security using Petri nets to model workflows. However, [3] does not consider the deployment of resources within a workflow across a set of computational resources. In [4], the author proposed an extended Petri net formalism — information flow security nets (IFSNs) — to provide a way of modelling information flow security policies expressed through the net structure. In [5], the author proposed a security model derived from the IFSNs — security coloured Petri nets (SCPNs) — providing more compact representation of systems and supporting more efficient analyses of information flow. In [1], the author proposed to partition workflows over a set of available clouds in such a way that security requirements are met. This approach was based on a multi-level security model extending Bell-LaPadula [2], [6] to encompass cloud computing. In [1], the author investigated workflow transformations that are needed when data is communicated between clouds, but did not consider the concurrency in the execution of tasks nor the opacity properties of a system. In [7], [8], [9], [10], the authors developed workflows for a cloud-based platform, where workflow is considered to be a linked set of individual components (*blocks*) which act sequentially upon items of data. However, the information flow security and opacity of the system was not considered.

There are different methods aimed at flow-sensitive analysis of programs. In [11], the authors introduced a general concept of security policy and a rigorous treatment of intransitive information flow. In [12], the author first showed that the data manipulated by a program can be constrained with security levels which usually have the structure of a partially ordered set. Moreover, this partially ordered set is a lattice under certain natural conditions [13]. In [14], the authors presented a lattice-based framework for both information and flow. In [15], the author first built a formal correspondence between mutual information and non-interference, and established a connection between Shannon's information theory and state-machine models of

- *Wen Zeng is with Cyber Security Centre, School of Computer Science and Informatics, De Montfort University, Leicester LE1 9BH, U.K. E-mail: wen.zeng.wz@gmail.com*
- *Maciej Koutny is with School of Computing, Newcastle University, Newcastle upon Tyne NE1 7RU, U.K. E-mail: maciej.koutny@ncl.ac.uk*

information flow in computing system. In [16], the authors devised a new information theoretic definition of information flow and channel capacity. In [17], the authors presented a notion of soundness for system that can be viewed as a form of non-interference. In [18], the authors proposed that Shannon's information theory be used to measure the information leakage in imperative programming languages.

Information hiding systems are used to formally analyse the information-hiding properties of protocols and programs. In [19], [20], [21], the authors defined generalised opacity for Petri nets, and then adapted opacity to the general transition system model. [22], [23] proposed model checking techniques to compute the Shannon entropy leakage and the min-entropy leakage in probabilistic transition systems. [24], [25] studied the problem of information hiding in systems characterized by the presence of randomization and concurrency. [26], [27], [28], [29] used diagnosis to detect whether or not the given sequence of observed labels indicates that some unobservable fault has occurred.

In this paper, we work towards bridging a gap between the theory of opacity and its practical application in a probabilistic setting. The study presented in this paper can help organizations to analyse the impact of different resource allocation strategies. It can also provide help to cloud providers in making security related decision at the system design stage.

The paper is organized as follows. Section 2 provides the basic notions used throughout this paper, and Section 3 a model for secure information flow analysis is presented. Opacity is discussed in Section 4, and a threat model is introduced in Section 5. Section 6 proposes different methodologies to quantify opacity. A cost function is given to help service providers take decisions related to information security in Section 7.

2 Preliminary Material

In this part, the definition of the basic concepts and key notions are presented making it easier to follow the technical content of this paper.

In order to measure the information flow, the system is treated as a communication channel. Information theory introduced the definition of entropy $\mathcal{H}$, to measure the average uncertainty in random variables. Shannon's measures were based on a logarithmic measure of the unexpectedness in a probabilistic settting. The unexpectedness of an event, which occurred with some non-zero probability p is $\log_2 \frac{1}{p}$. Therefore, the total information carried by a set of n events is computed as the weighted sum of their unexpectedness:

$$\mathcal{H} = \sum_{i=1}^{n} p_i \log_2 \frac{1}{p_i} \ , \tag{1}$$

where p_i is the probability of event i, and $p_i \log_2 \frac{1}{p_i} = 0$ if $p_i = 0$. (1) is called the entropy of the set of events.

A discrete random variable X is a surjective function, which maps events to values of a countable set, with each value in the range having probability greater than zero, i.e., $X : D \to R$, where D is a finite set with a specified probability distribution, and R is the finite range of X.

Shannon's information theory [30] can be used to quantify the amount of information a system may leak and the way in which this depends on the distribution of inputs.

Definition 1 (Shannon entropy)**.** *Let X be a random variable, x range over the set of values which X may take, and $p(x)$ denotes the probability that X takes the value x. The* entropy *of a discrete random variable X is denoted by $\mathcal{H}(X)$ and is defined by:*

$$\mathcal{H}(X) = \sum_{x} p(x) \log_2 \frac{1}{p(x)} \ . \tag{2}$$

The entropy measures the average information content of a set of events.

Definition 2 (conditional entropy)**.** *The* conditional entropy *$\mathcal{H}(X|Y)$ measures the uncertainty about X, given the knowledge of $Y = y$. It is defined as:*

$$\begin{aligned} \mathcal{H}(X|Y) &= \textstyle\sum_y p(y)\mathcal{H}(X|Y=y) \\ &= \mathcal{H}(X,Y) - \mathcal{H}(Y) \ , \end{aligned} \tag{3}$$

where $\mathcal{H}(X|Y=y)$ is the entropy of the discrete random variable X conditioned on the discrete random variable Y taking a certain value y. The joint entropy $\mathcal{H}(X,Y)$ of a pair of discrete random variables (X,Y) with a joint distribution $p(x,y)$ is defied as:

$$\mathcal{H}(X,Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \frac{1}{\log_2 p(x,y)} \ . \tag{4}$$

Given two random variables, X and Y, the conditional entropy captures dependencies between random variables, when the knowledge of one may change the information about the other. If $\mathcal{H}(X|Y) = 0$, there is no uncertainty on X knowing Y; and if X and Y are independent (i.e., $p(x,y) = p(x)p(y)$), then $\mathcal{H}(X|Y) = \mathcal{H}(X)$, meaning that the knowledge of Y does not change the uncertainty on X.

The concept of mutual information is a measure of the amount of information that one random variable contains about another random variable. It implies the reduction in the uncertainty of one random variable due to the knowledge of the other.

Definition 3 (mutual information)**.** *Let $p(x,y)$ be the joint distribution of two random variables, X and Y. The* mutual information *$\mathcal{I}(X;Y)$ between X and Y is given by:*

$$\begin{aligned} \mathcal{I}(X;Y) &= \textstyle\sum_x \sum_y p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)} \\ &= \mathcal{H}(X) - \mathcal{H}(X|Y) = \mathcal{H}(Y) - \mathcal{H}(Y|X) \ . \end{aligned} \tag{5}$$

If X and Y are independent, then knowing X does not reveal any information about Y and vice versa, so their mutual information is zero. At the other extreme, if X and Y are identical then all information conveyed by X is shared with Y. As a result, in the case of identity, the mutual information is the same as the uncertainty contained in X (or Y) alone, namely the entropy of X (or Y as identical X and Y have equal entropy).

3 System Model

In this section, we introduce a formal model for capturing information flow in federated cloud systems.

In the context of a federated cloud system, we assume that there exists a finite nonempty set $\mathcal{C}$ of single deployment clouds. Moreover, $\mathcal{E}$ is a finite nonempty set of *entities*, which comprise both subjects (e.g., services and programs), and objects (e.g., resources and messages).

An entity can have several copies, and each of these copies may reside in a different cloud. We also allow multiple copies of an entity to be present in a single deployment cloud. As a result, in what follows, a *system state* will be any finite multiset st over the Cartesian product $\mathcal{E} \times \mathcal{C}$. Thus, for example, $st(a,c) = 4$ means that in the current state st there are 4 copies of entity a residing in cloud c. The elements of $\mathcal{E} \times \mathcal{C}$ will be referred to as *actual* entities. We will say that an actual entity (e,c) is *present* in state st if $st(e,c) > 0$.

Definition 4 (PIFM)**.** *A* probabilistic information flow model *is a triple:*

$$PIFM = (\mathcal{A}, st_{init}, \mu) \ , \tag{6}$$

where $\mathcal{A}$ is a finite set of actions, *st_{init} is an* initial state, *and $\mu : \mathcal{A} \to \mathbb{N}$ is a* probability weight *mapping. It is assumed that each action is a pair:*

$$\phi = (in, out) \tag{7}$$

such that its two components, $in = ((e_1,c_1),\ldots,(e_k,c_k))$ and $out = ((e_{k+1},c_{k+1}),\ldots,(e_{k+m},c_{k+m}))$, are nonempty finite tuples of actual entities.

One may extend the class of allowed action types to include, for example, features allowing checking of the absence of certain actual entities.

Definition 5 (actions)**.** *An action ϕ as in (7) is* enabled *at state st if:* [1]

$$\phi^{in} = \{(e_1,c_1),\ldots,(e_k,c_k)\} \le st, \tag{8}$$

and we denote by enabled(st) the set of all actions enabled in a state st. An enabled action ϕ can be executed *with the probability:*

$$pr = \frac{\mu(\phi)}{\sum_{\phi' \in enabled(st)} \mu(\phi')} \tag{9}$$

leading to a new state $st' = st - \phi^{in} + \phi^{out}$, where:

$$\phi^{out} = \{(e_{k+1},c_{k+1}),\ldots,(e_{k+m},c_{k+m})\} \ . \tag{10}$$

We denote this by $st \xrightarrow{\phi}_{pr} st'$.

Among the possible system states, one is only interested is those which can be reached from the initial state.

Definition 6 (reachable states)**.** *The set of* reachable states *of the probabilistic information flow model $PIFM$ as in (6) is the minimal set of states RS containing st_{init} and such that if $st \in RS$ and $st \xrightarrow{\phi}_{pr} st'$, for some ϕ and pr, then $st' \in RS$.*

We presented basic notions related to the syntax and semantics of a probabilistic information flow model. It allows a straightforward capture of various notions related to the information flow in federated cloud systems.

The model introduced in this section resembles Petri net models proposed in [31], [32] which incorporated the Bell-Lapadula rules in cloud computing systems.

4 Opacity in Cloud Computing Systems

In cloud computing, observing patterns of users' behaviour can lead to leakages of secure information. Information sharing means that the behaviour of one cloud user may appear visible to other cloud users or adversaries, and observations of such behaviours can potentially help them to build covert channels. It is, therefore, necessary to reason about information leakage in a formal and precise way.

We consider using opacity as a promising technique for analyzing information flow security. Opacity has been proposed as a uniform approach for describing security properties of computing systems expressed as predicates [19], [20], [21]. A predicate is opaque if an observer of the system is unable to determine the truth of the predicate in a given system run. In this section, we will present one of the versions of opacity which can be used, e.g., to analyze workflows executed in cloud computing systems.

Definition 7 (runs)**.** *A* run *of a probabilistic information flow model $PIFM$ as in (6) is a finite sequence actions in $\mathcal{A}$:*

$$\xi = \phi_1 \ldots \phi_n \tag{11}$$

such that there are states $st_{init} = st_1,\ldots,st_{n+1}$ satisfying:

$$st_1 \xrightarrow{\phi_1}_{pr} st_2 \xrightarrow{\phi_2}_{pr} \cdots \xrightarrow{\phi_{n-1}}_{pr} st_n \xrightarrow{\phi_n}_{pr} st_{n+1} \ . \tag{12}$$

The set of all runs of $PIFM$ is denoted by $Runs(PIFM)$.

A variety of different observing capabilities of the behaviours of the system modelled by $PIFM$ can be captured by observation functions.

Definition 8 (observations)**.** *An* observation function *of a probabilistic information flow model $PIFM$ as in (6) is:*

$$obs : Runs(PIFM) \to Obs^* \ , \tag{13}$$

where Obs is a set of observables. *Moreover, obs is* static[2] *if there is a mapping $obs' : \mathcal{A} \to Obs \cup \{\epsilon\}$ such that:*

$$obs(\xi) = obs'(\phi_1) \ldots obs'(\phi_n) \ , \tag{14}$$

for every run $\xi = \phi_1 \ldots \phi_n$ in $Runs(PIFM)$.

Note that obs' may return the empty sequence, ϵ, which means that one can model invisible (or hidden) actions.

Given an observation function obs, we are interested in whether an observer can establish a property γ (a predicate over system runs) for a run of $PIFM$ having only access to the result of the observation function. As one can identify γ with its characteristic set, i.e., the set of all those runs for which it holds, we would want to find out whether the fact that the underlying run belongs to $\gamma \subseteq Runs(PIFM)$ can be deduced by the observer on the basis of an observed execution of the system. In particular, we might be interested in the *final* opacity predicate γ_Z, defined as the set of all the runs ξ as in (11) satisfying $st_n \in Z$, for some set of states Z [21]. Intuitively, this means that might be interested in finding out whether the observation of a run carries sufficient information to deduce that the latter ended in a state of Z.

[1] Below $\le$ denotes multiset inclusion, and '$-$' and '$+$' denote multiset subtraction and addition, respectively.

[2] Static in the sense that a given action is always observed in the same way.

Note that we are not interested in establishing whether the underlying run does not belong to γ; to do this, we would consider the property:

$$\overline{\gamma} = Runs(PIFM) \setminus \gamma \ . \tag{15}$$

Definition 9 (opacity)**.** *Let $PIFM$ be a probabilistic information flow model as in (6), and obs be an observation function as in (13).*

A set of runs $\gamma \subseteq Runs(PIFM)$ is opaque *w.r.t. the observation function obs if, for every run $\xi \in \gamma$, there is a run $\xi \in obs(\overline{\gamma})$ such that $obs(\xi) = obs(\xi')$.*

In other words, γ is opaque if all the runs in γ are covered by runs in $\overline{\gamma}$:

$$obs(\gamma) \subseteq obs(\overline{\gamma}) \ . \tag{16}$$

4.1 Case study

As a case study adapted to our purposes, we will use a medical research application [1] in which data from a set of patients' heart rate monitors is analyzed, and then the results are sent to the things (i.e., computing devices, digital machines, objects, or people), as illustrated in Figure 1. Informally, the process can be described as follows:

- A patients' data is sent through the local network by $user$. The data (d_0) is a file with a header identifying the patient (name and patient number), followed by a set of heart rate data recoded over a period of time.
- A service ($serv{:}s_0$) reads the data, and changes the name of the data to (d_1), then sends the data to service ($serv{:}s_1$).
- A service ($serv{:}s_1$) strips off the header, leaving only the heart rate data (d_2).
- Another service ($serv{:}s_2$) analyzes the heart rate data, and produces results (d_3).
- Finally, service ($serv{:}s_3$) sends the data (d_4) to $user/medical\ device$, and then a 'medical device' can provide appropriate service for the patient based on the received data.

Analyzing the heart rate data ($serv{:}s_3$) is costly and would benefit from a cheap, scalable resources that are available on public clouds. However, considering that storing medical records on a public cloud can breach confidentiality, some organizations prefer to deploy the whole workflow on a secure private cloud. Such a policy may overstretch the limited resources available on the private cloud, resulting in degraded performance and negative impact on other applications. To address this problem, the partitioning of the application between a private cloud and a public cloud could provide a better solution.

In our case study, we consider an integration of internet of things (IoT) with cloud computing. We use two clouds (X and Y), one local network LN, and a number of processes, which together form a *medical research* application. The proposed workflow operates on sensitive medical data processed on the private cloud, and anonymised data that can be deployed on the public cloud. Cloud X hosts services $serv{:}s_0$ and $serv{:}s_1$. Cloud Y hosts two service providers: *service provider 1* includes $serv{:}s_2^1$ and $serv{:}s_3^1$; and *service provider 2* includes $serv{:}s_2^2$ and $serv{:}s_3^2$.

TABLE 1
The sequence of interactions between the components of the medical research application system.

Actions	Entities	Sender	Receiver
ϕ_1	d_0	$user$	$serv{:}s_0$
ϕ_2	d_1	$serv{:}s_0$	$serv{:}s_1$
ϕ_3^i	d_2	$serv{:}s_1$	$serv{:}s_2^i$ (i = 1,2)
ϕ_4^i	d_3	$serv{:}s_2^i$ (i = 1,2)	$serv{:}s_3^i$ (i = 1,2)
ϕ_5^i	d_4	$serv{:}s_3^i$ (i = 1,2)	$user$
ϕ_6^i	d_4	$serv{:}s_3^i$ (i = 1,2)	$medical\ device$

Figure 2 shows the basic structure of the execution scenario for the medical research application, and its generic behaviour is shown in Table 1. It starts with a data sent from $user$ to $serv{:}s_0$, through a local network. The data is forwarded to service $serv{:}s_1$. Service $serv{:}s_1$ then selects one of the two providers, *service provider 1* or *service provider 2*, and then sends d_2 to the selected service providers. After receiving the data, $serv{:}s_2^1$ or $serv{:}s_2^2$ produces d_3 and sends it to $serv{:}s_3^1$ or $serv{:}s_3^2$, respectively. Finally, $serv{:}s_3^1$ and $serv{:}s_3^2$ sends d_4 to $user$ and $medical\ device$. Crucially, an external observer of the system is not allowed to discover the identity of the selected provider.

Scenario A

We assume that no provider is discriminated against. Messages communicated between the clouds X, Y and local network are visible, and messages inside the clouds are invisible. Moreover, the observer has no means of detecting their content (but can observe the specific cloud from which a message originated or was sent to). This can be captured by a static observation function obs given by:

$$\begin{array}{lclclcl} obs'(\phi_1) & = & a & \quad & obs'(\phi_2) & = & \epsilon \\ obs'(\phi_3^1) & = & b & & obs'(\phi_3^2) & = & b \\ obs'(\phi_4^1) & = & \epsilon & & obs'(\phi_4^2) & = & \epsilon \\ obs'(\phi_5^1) & = & d & & obs'(\phi_5^2) & = & d \\ obs'(\phi_6^1) & = & e & & obs'(\phi_6^2) & = & e \ . \end{array} \tag{17}$$

We can formulate as an opacity problem the question of whether visible interactions reveal the identity of the provider supplying the service. To do so, we consider a property γ consisting of all runs where the first provider supplied the services, i.e., executions of the following form: [3]

$$\begin{array}{lcl} \xi_1 & = & \phi_1\phi_2\phi_3^1\phi_4^1\phi_5^1\phi_6^1 \\ \xi_2 & = & \phi_1\phi_2\phi_3^1\phi_4^1\phi_6^1\phi_5^1 \ . \end{array} \tag{18}$$

The set of observations generated by γ is therefore given by $obs(\gamma) = \{obs(\xi_i) \mid i = 1, 2\}$, where:

$$\begin{array}{lcl} obs(\xi_1) & = & abde \\ obs(\xi_2) & = & abed \ . \end{array} \tag{19}$$

We then note that $\overline{\gamma}$ comprises, among others, executions of the following kind:

$$\begin{array}{lcl} \xi_1 & = & \phi_1'\phi_2\phi_3^2\phi_4^2\phi_5^2\phi_6^2 \\ \xi_2 & = & \phi_1'\phi_2\phi_3^2\phi_4^2\phi_6^2\phi_5^2 \ . \end{array} \tag{20}$$

[3] In the examples, we only consider complete runs.

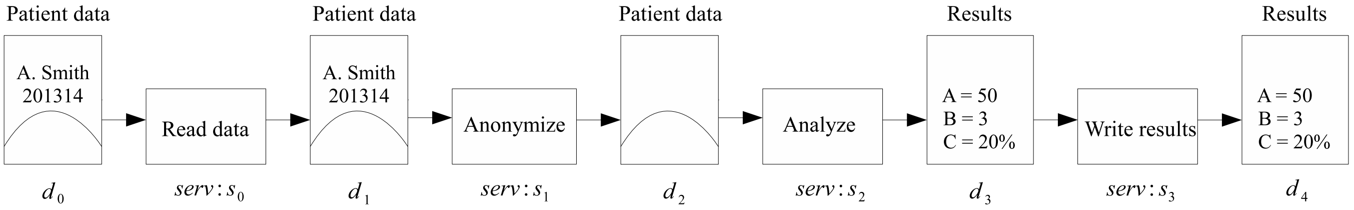

Fig. 1. The medical research application example, which includes four services and five finds of data.

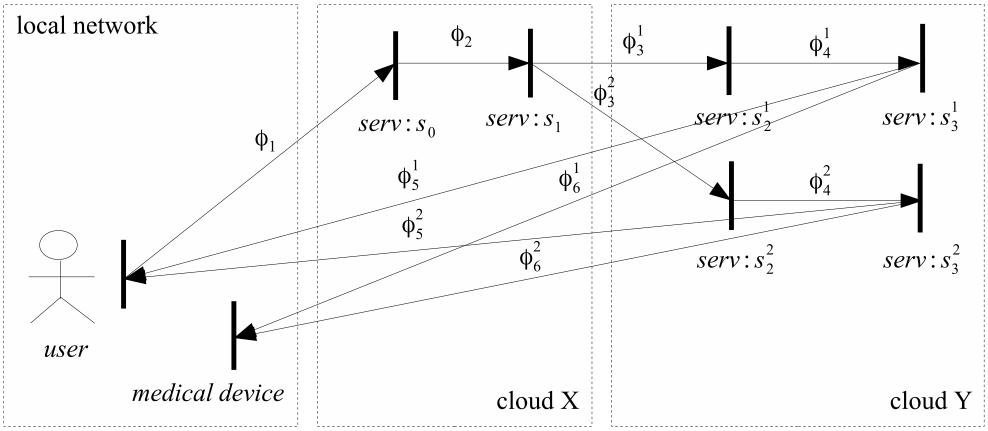

Fig. 2. Information flow in a cloud based *medical research* application.

The set of observations generated therefore satisfies $obs(\overline{\gamma}) \supseteq \{obs(\xi_i) \mid i = 1, 2\}$, where:

$$\begin{array}{rcl} obs(\xi'_1) & = & abde \\ obs(\xi'_2) & = & abed\ . \end{array} \tag{21}$$

Hence $obs(\gamma) \subseteq obs(\overline{\gamma})$, and so γ is an opaque property in this case. As a result, it is not possible to say for sure that it was the first provider who supplied the service. Since the above argument is symmetric, we can conclude that the identity of providers is kept secret.

Scenario B

We assume that cloud Y is public and cloud X is private. Messages communicated between the clouds X, Y and the local network are visible, messages inside Y are visible, messages inside X are invisible. This can be captured by a static observation function obs given by:

$$\begin{array}{rclrcl} obs'(\phi_1) & = & a & obs'(\phi_2) & = & \epsilon \\ obs'(\phi_3^1) & = & b & obs'(\phi_3^2) & = & c \\ obs'(\phi_4^1) & = & d & obs'(\phi_4^2) & = & e \\ obs'(\phi_5^1) & = & f & obs'(\phi_5^2) & = & g \\ obs'(\phi_6^1) & = & h & obs'(\phi_6^2) & = & k\ . \end{array} \tag{22}$$

We consider the property γ consisting of all execution scenarios where the first provider supplied the services (the executions are the same as in Scenario A). The set of observations they generate is given by $obs(\gamma) = \{obs(\xi_i) : i = 1, 2\}$, where:

$$\begin{array}{rcl} obs(\xi_1) & = & abdfh \\ obs(\xi_2) & = & abdhf\ . \end{array} \tag{23}$$

When the second provider supplied the services, the set of observations generated is $obs(\overline{\gamma}) = \{obs(\xi'_i) \mid i = 1, 2\}$, where:

$$\begin{array}{rcl} obs(\xi'_1) & = & acegk \\ obs(\xi'_2) & = & acekg\ . \end{array} \tag{24}$$

Hence $obs(\gamma) \setminus obs(\overline{\gamma}) \neq \emptyset$, and so γ is not an opaque property. Thus, it is now possible to say that it was the first provider who supplied the services.

From these two scenarios we can see that an observation cannot establish a predicate if for any run of the system in which the predicate is true, there is a run for which the predicate is false, and the two runs are equivalent under the defined observation function.

In this section, we only considered how security policies can affect the information flow security. We have not yet taken into account the likelihood of violating the opacity requirement; instead, we simply reported whether a given system is opaque or not. This yes/no outcome might be not satisfactory in practice, especially when system behaviours have unequal likelihood of occurring. In the case where the probability of one run is significantly higher than the probability of another one, the observer may have good reasons to believe that the predicate is nonetheless true. Therefore, in the next section, we will consider the probabilistic behaviour of the system.

5 Threat Model

We now consider the probabilistic opacity in the cloud computing systems, which allows us to reason about the quantitative properties of systems.

Consider that a malicious service provider (insider) can observe users' behaviours and interactions with the server by studying the patterns of users behaviours in a cloud computing system. In such a case, the malicious insider can deduce information about the users or service providers, which might cause confidential information leakage [33], [34]. For example, in [7], [8], workflows are developed in a cloud-based platform, where a workflow is a linked set of individual components (*blocks*) which act sequentially upon items of data.

For the purpose of measuring opacity, we consider probability distributions on random variables of user behaviour runs. We consider the system to be a communication channel, and a discrete random variable is used to model a finite set of possible runs performed by the end users during their interactions with the system.

Definition 10. *A* service-based computing system *is a tuple:*

$$sbcs = (\mathcal{U}, \mathcal{A}, Obs, \mathcal{R}, \mathcal{O}, \mu_{\mathcal{R}}, \mu_{\mathcal{O}}) \ , \qquad (25)$$

where:

- $\mathcal{U}$ *is a finite set of* users *interacting with the system.*
- $\mathcal{A}$ *is a finite set of* actions *modelling the interactions between the system and the users, see Definition 4.*
- Obs *is a finite set of* observables *and* $obs : \mathcal{A}^* \to Obs^*$.
- $\mathcal{R}$ *is a finite set of* actual runs *modelling interactions of the users with the system.*
- $\mathcal{O}$ *is a finite set of* observed runs *obtained from the actual runs using an observation function* obs.
- $\mu_{\mathcal{R}}$ *is a probability distribution on the actual runs.*
- $\mu_{\mathcal{O}}$ *is a probability distribution on the observed runs.*

Each run is a finite sequence of actions performed by the end user, $\xi = \phi_1 \ldots \phi_k$, which captures the way in which an end user reaches the final action ϕ_k from the initial action ϕ_1 when requesting a specific service. All possible runs of a user requesting a service located in the system be denoted by:

$$\mathcal{R} = \{\xi_j \mid 1 \leq j \leq n\} \ \ and \ \ \sum_{i=1}^{n} \mu_{\mathcal{R}}(\xi_j) = 1 \ . \qquad (26)$$

Following the definition of *observation function* in Section 4, we have a number of distinct observed runs, $\psi_1, \ldots, \psi_m$, such that $\{\psi_1, \ldots, \psi_m\} = obs(\{\xi_1, \ldots, \xi_n\})$. The probability of each observed trace ψ_i is $\mu_{\mathcal{O}}(\psi_i) = \sum_{\xi_j \in obs^{-1}(\psi_i)} \mu_{\mathcal{R}}(\xi_j)$, and so we have:

$$\mathcal{O} = \{\psi_i \mid 1 \leq i \leq m\} \ \ and \ \ \sum_{i=1}^{m} \mu_{\mathcal{O}}(\psi_i) = 1 \ . \qquad (27)$$

We assume that there are two types of actions: *hidden* actions and *observable* actions, which are related to *high* security level and *low* security level, respectively. Actions labelled *hidden* are confidential and hidden to the adversaries, and actions labelled *observable* are public and observable to the adversaries. The classification can be based on the security preserving mechanisms or policies applied in the computing system. Therefore, actions are the union of two disjoint sets:

$$\mathcal{A} = high \uplus low \qquad (28)$$

For each run, some part of it is hidden and some part of it is observable, which makes some runs equivalent to the others when only considering observable actions. Therefore, adversaries can derive some confidential information by building sets of equivalence classes from observations.

We now look at the properties of observations on the behaviours of users within the computing system through equivalence relations (in the examples hidden actions will be underlined).

Example 1. *Consider Scenario A in Section 4.1, there are four different actual runs that can be generated by the system* $\mathcal{R} = \{\xi_1, \xi_2, \xi_3, \xi_4\}$ *and we assume* $\mu_{\mathcal{R}}(\xi_i) = p_i$*, we have:*

- $\xi_1 = \phi_1 \underline{\phi_2} \phi_3^1 \underline{\phi_4^1} \phi_5^1 \phi_6^1 \qquad \to p_1$
- $\xi_2 = \phi_1 \underline{\phi_2} \phi_3^1 \underline{\phi_4^1} \phi_6^1 \phi_5^1 \qquad \to p_2$
- $\xi_3 = \phi_1 \underline{\phi_2} \phi_3^2 \underline{\phi_4^2} \phi_5^2 \phi_6^2 \qquad \to p_3$
- $\xi_4 = \phi_1 \underline{\phi_2} \phi_3^2 \underline{\phi_4^2} \phi_6^2 \phi_5^2 \qquad \to p_4$

Moreover, $\sum_{i=1}^{4} p_i = 1$*. The actual runs are obfuscated due to the applied security policy of the system which produces observable runs. Since the underlined actions,* ϕ_2*,* ϕ_4^1 *and* ϕ_4^2 *are* high *security and so hidden, we obtain the following observations from an attacker's point of view:*

- $\delta_1 = abde \qquad \to \mu_{\mathcal{O}}(\delta_1)$
- $\delta_2 = abed \qquad \to \mu_{\mathcal{O}}(\delta_2)$

and $\mathcal{O} = \{\delta_1, \delta_2\}$*. In other words, the observation function can be summarised as:*

- $obs(\xi_1) = obs(\xi_3) = \delta_1$
- $obs(\xi_2) = obs(\xi_4) = \delta_2$

The probability of observing δ_1 *is* $\mu_{\mathcal{O}}(\delta_1) = p_1 + p_3$*, and the probability of observing* δ_2 *is* $\mu_{\mathcal{O}}(\delta_2) = p_2 + p_4$*. The adversary therefore builds equivalence relations on the inverse images of the observations and derives information from the observations of users' behaviours.*

In the above example, two runs are equivalent if they have the same observation. Intuitively, we consider the observation as the sum of probability distribution for the projections of actual runs on visible actions. Information about users' behaviours can only be partially deduced.

6 Quantitative Analysis of Opacity

In this section, we introduce different variants of quantitative opacity for cloud computing systems.

6.1 Observational Equivalence: π_ρ-opacity

If ξ_1 and ξ_2 are two actual runs with the same observations, i.e., $obs(\xi_1) = obs(\xi_2)$, we denote $\xi_1 \backsimeq \xi_2$ and say that they are *observationally equivalent*.

Definition 11. *A predicate* $\gamma \subset \mathcal{R}$ *is* π_ρ-opaque *w.r.t.* obs *if the probability of having a run in* γ *which is not covered by a run outside* γ *is* ρ*:*

$$\rho = \frac{1}{\mu_{\mathcal{R}}(\gamma)} \cdot \mu_{\mathcal{R}}(\{\xi \in \gamma \mid obs(\xi) \in obs(\mathcal{R} \backslash \gamma)\}) \ . \qquad (29)$$

Example 2. *Consider Scenario A in Section 4.1. The actual runs are* $\mathcal{R} = \{\xi_1, \ldots, \xi_4\}$*. In addition, we assume that messages communication between the clouds* Y *and local network, i.e.,* ϕ_5^1*,* ϕ_5^2*,* ϕ_6^1*, and* ϕ_6^2*, are of high security and hidden, where:*

- $\xi_1 = \phi_1 \underline{\phi_2} \phi_3^1 \underline{\phi_4^1 \phi_5^1 \phi_6^1} \qquad \to \frac{1}{4}$

- $\xi_2 = \phi_1\underline{\phi_2}\phi_3^1\underline{\phi_4^1\phi_6^1\phi_5^1} \qquad \rightarrow \frac{1}{2}$
- $\xi_3 = \phi_1\underline{\phi_2}\phi_3^2\underline{\phi_4^2\phi_5^2\phi_6^2} \qquad \rightarrow \frac{1}{8}$
- $\xi_4 = \phi_1\underline{\phi_2}\phi_3^2\underline{\phi_4^2\phi_6^2\phi_5^2} \qquad \rightarrow \frac{1}{8}$

Then we have:

$$obs(\xi_1) = obs(\xi_2) = obs(\xi_3) = obs(\xi_4) = ab \tag{30}$$

i.e., $\xi_1 \overset{\frown}{-} \xi_2 \overset{\frown}{-} \xi_3 \overset{\frown}{-} \xi_4$*. Hence all the runs in* $\mathcal{R}$ *are observationally equivalent, and so each* $\gamma \subset \mathcal{R}$ *is* π_1*-opaque.*

Note that one could easily associate with each run $\xi \in \gamma$ a numerical weight κ_ξ, reflecting its relative *importance* or *security level*. Then (29) could be replaced by:

$$\rho = \frac{1}{\nu(\gamma)} \cdot \nu(\{\xi \in \gamma \mid obs(\xi) \in obs(\mathcal{R} \backslash \gamma)\}) \ , \tag{31}$$

where $\nu(X) = \sum_{\xi \in X} \kappa_\xi \cdot \mu_{\mathcal{R}}(\xi)$. The same comment applies to the opacity measure introduced next.

6.2 Proportion-based Opacity: $\widetilde{\pi}_\chi$-opacity

The observational equivalence may be too demanding in practice. In particular, one might only require that the probability of not covered runs is low. To capture this measure, we first define a notion of quantified opacity as the proportion of the equivalent runs w.r.t. the whole set of actual runs. The higher such proportion is, the more secure the system becomes. This also corresponds to an intuition that a lower probability of non-equivalent runs means that it is harder for an attacker to find differences in users' behaviours.

Definition 12. *The opacity* proportion *measure is the probability of the set of the covered runs:*

$$\chi = \mu_{\mathcal{R}}(\{\ \xi \in \mathcal{R} \mid obs(\xi) \in obs(\mathcal{R} \backslash \{\xi\})\}) \ . \tag{32}$$

We also say that the runs $\mathcal{R}$ *are* $\widetilde{\pi}_\chi$-opaque.

The opacity proportion is maximal if $\chi = 1$, because an attacker cannot identify the actual runs behind the observed ones. It is minimal if $\chi = 0$.

Example 3. *Consider again the example in Section 4.1. In Scenario A, there are four different actual runs that can be generated by the system* $\mathcal{R} = \{\xi_1, \ldots, \xi_4\}$ *and:*

- $\xi_1 = \phi_1\underline{\phi_2}\phi_3^1\underline{\phi_4^1}\phi_5^1\phi_6^1 \qquad \rightarrow \frac{1}{8}$
- $\xi_2 = \phi_1\underline{\phi_2}\phi_3^1\underline{\phi_4^1}\phi_6^1\phi_5^1 \qquad \rightarrow \frac{1}{4}$
- $\xi_3 = \phi_1\underline{\phi_2}\phi_3^2\underline{\phi_4^2}\phi_5^2\phi_6^2 \qquad \rightarrow \frac{1}{8}$
- $\xi_4 = \phi_1\underline{\phi_2}\phi_3^2\underline{\phi_4^2}\phi_6^2\phi_5^2 \qquad \rightarrow \frac{1}{2}$

The observation function yields:

- $obs(\xi_1) = obs(\xi_3) = abde \qquad \rightarrow \frac{1}{4}$
- $obs(\xi_2) = obs(\xi_4) = abed \qquad \rightarrow \frac{3}{4}$

Hence, the opacity proportion is $\chi = 1$*, and the runs* $\mathcal{R}$ *are* $\widetilde{\pi}_1$*-opaque. We also have that* $\gamma = \{\xi_1, \xi_2, \xi_3\}$ *is* $\pi_{0.25}$*-opaque.*

In Scenario B, $\mathcal{R} = \{\xi_1, \ldots, \xi_4\}$ *and we have:*

- $\xi_1 = \phi_1\underline{\phi_2}\phi_3^1\phi_4^1\phi_5^1\phi_6^1 \qquad \rightarrow \frac{1}{8}$
- $\xi_2 = \phi_1\underline{\phi_2}\phi_3^1\phi_4^1\phi_6^1\phi_5^1 \qquad \rightarrow \frac{1}{4}$
- $\xi_3 = \phi_1\underline{\phi_2}\phi_3^2\phi_4^2\phi_5^2\phi_6^2 \qquad \rightarrow \frac{1}{8}$
- $\xi_4 = \phi_1\underline{\phi_2}\phi_3^2\phi_4^2\phi_6^2\phi_5^2 \qquad \rightarrow \frac{1}{2}$

The observation function yields:

- $obs(\xi_1) = abdfh \qquad \rightarrow \frac{1}{8}$
- $obs(\xi_2) = abdhf \qquad \rightarrow \frac{1}{4}$
- $obs(\xi_3) = acegk \qquad \rightarrow \frac{1}{8}$
- $obs(\xi_4) = acekg \qquad \rightarrow \frac{1}{2}$

Therefore, the opacity proportion is $\chi = 0$*, and the runs* $\mathcal{R}$ *are* $\widetilde{\pi}_0$*-opaque.*

6.3 Entropy-based Opacity: $\overline{\pi}_{\mathcal{I}}$-opacity

The measurement we consider here is based on the likelihood of a particular run of actions being performed by the end user, i.e., on the probability distribution on the user's behaviours to measure the observations. Shannon's measures are based on a logarithmic measure of uncertainty, inherent in a probabilistic event. Therefore, it is natural to consider Shannon's entropy as the basis of the information loss measurement.

We consider possible runs of an end user interacting with a service provider in the system as a random variable, and define a notion of quantified opacity by using the concept of mutual information between the actual runs and their observations.

Definition 13. *Consider a user* u *requesting a service. The quantity of entropy-based information (uncertainty) loss due to user* u *in the system (25) is:*

$$\mathcal{I} = \mathcal{I}(T_u; O_u) = \mathcal{H}(T_u) - \mathcal{H}(T_u|O_u) \tag{33}$$

where: (i) T_u *and* O_u *are the random variables for the user* u*'s actual runs* $\mathcal{R}$ *and observations runs* $\mathcal{O}$*; (ii)* $\mathcal{H}(T_u)$ *is the entropy (uncertainty measurement) of runs in* $\mathcal{R}$*; and (iii)* $\mathcal{H}(T_u|O_u)$ *is the conditional entropy of* T_u*, given the observation* O_u *(the remaining uncertainty* $\mathcal{R}$ *after observing* $\mathcal{O}$*). We also say that the user's behaviour is* $\overline{\pi}_{\mathcal{I}}$-opaque *where* $\mathcal{I} = \mathcal{I}(T_u; O_u)$*.*

Example 4. *Consider again the example in Section 4.1. There are four different actual runs that can be generated by the system and we assume* $\mu_{\mathcal{R}}(\xi_i) = \frac{1}{4}$*, for all* i*.*

In Scenario A, there are two different observed runs (δ_1*,* δ_2*), and we have:* $\mu_{\mathcal{O}}(\delta_1) = \mu_{\mathcal{O}}(\delta_2) = \frac{1}{2}$*. For* $\mathcal{R} = \{\xi_1, \xi_2, \xi_3, \xi_4\}$ *and* $\mathcal{O} = \{\delta_1, \delta_2\}$*, we then obtain:*

$$\begin{aligned} \mathcal{I}(\mathcal{R}; \mathcal{O}) &= \mathcal{H}(\xi) - \mathcal{H}(\xi|\delta) \\ &= 4 \cdot \tfrac{1}{4}\log_2 4 - (2 \cdot \tfrac{1}{2}\log_2 2) = 1 \ . \end{aligned} \tag{34}$$

The mutual information between $\mathcal{R}$ *and* $\mathcal{O}$ *is therefore 1, yielding* $\overline{\pi}_1$*-opacity.*

In Scenario B, there are four different observed runs, (δ_1*,* δ_2*,* δ_3*,* δ_4*), and we have:* $\mu_{\mathcal{O}}(\delta_1) = \mu_{\mathcal{O}}(\delta_2) = \mu_{\mathcal{O}}(\delta_3) = \mu_{\mathcal{O}}(\delta_4) = \frac{1}{4}$*. We then obtain:*

$$\begin{aligned} \mathcal{I}(\mathcal{R}; \mathcal{O}) &= \mathcal{H}(\xi) - \mathcal{H}(\xi|\delta) \\ &= 4 \cdot \tfrac{1}{4}\log_2 4 - (4 \cdot \tfrac{1}{4}\log_2 4) = 0 \ . \end{aligned} \tag{35}$$

The mutual information between $\mathcal{R}$ *and* $\mathcal{O}$ *is 0, yielding* $\overline{\pi}_0$*-opacity.*

Note that since the observation of each run is different, the mutual information between $\mathcal{R}$ *and* $\mathcal{O}$ *is exactly the number of bits needed to encode 4 runs, i.e., 2. And the entropy (remaining uncertainty) is 0.*

IoT is a demanding environment due to the potentially unbounded number of things (resources and subjects). As the above discussion focused on the case of one end user interacting with the system, it is still necessary to define the security measurement for all end users in the system.

Definition 14. *The overall* quantity of the information loss *is the mean value of all users' information loss:*

$$\overline{\chi} = \sum_{i=1}^{m} p_i \cdot \mathcal{I}(T_i; O_i) \tag{36}$$

where: (i) m is the number of end users; (ii) p_i is the probability of all runs of user u_i requesting a service; (iii) T_i and O_i are random variables of actual and observed runs for user u_i accessing the service; and (iv) $\mathcal{I}(T_i; O_i)$ is the information loss of user u_i.

6.4 Channel Capacity: $\widehat{\pi}_C$-opacity

We now define opacity measurement based on the concept of channel capacity.

Let us consider the security preserving mechanisms providing secure communication channels for users and services in the information theoretical sense. The channel capacity is the maximum mutual information between T and O over all possible end users/entities w.r.t. to requesting a service, where T and O respectively denote the random variables of $\mathcal{R}$ (the set of actual runs) and $\mathcal{O}$ (the set of observed runs).

Definition 15. *The* channel capacity *of the computing system (25) is defined as:*

$$C = \max_{u \in \mathcal{U}} \mathcal{I}(T_u; O_u) \tag{37}$$

*when each user u requests a given service. We also say that the behaviour of the set of users is $\widehat{\pi}_C$-*opaque, *which is the maximal amount of information that can be obtained.*

The channel is *memoryless* if the probability distribution of the output depends only on the input and is conditionally independent of previous channel inputs and outputs.

Example 5. *Consider Scenario B in Section 4.1. We assume there are two end users u_1 and u_2 sending (ϕ_1) data d_0 to serv:s_0. The sum of the probabilities of all runs performed by all the users is 1. The probability of all possible runs of u_1 is $\frac{2}{3}$, and that of u_2 is $\frac{1}{3}$. The possible actual runs of each user and conditional probabilities of each user's runs are as follows:*

End Users	*Actual Traces*	*Probability*
$\frac{2}{3}.u_1$	$\xi_1 = \phi_1\phi_2\phi_3^1\phi_4^1\phi_5^1\phi_6^1$ $\xi_2 = \phi_1\phi_2\phi_3^1\phi_4^1\phi_6^1\phi_5^1$ $\xi_3 = \phi_1\phi_2\phi_3^2\phi_4^2\phi_5^2\phi_6^2$ $\xi_4 = \phi_1\phi_2\phi_3^2\phi_4^2\phi_6^2\phi_5^2$	$\frac{1}{6}$ $\frac{1}{6}$ $\frac{1}{3}$ $\frac{1}{3}$
$\frac{1}{3}.u_2$	$\xi_1 = \phi_1\phi_2\phi_3^1\phi_4^1\phi_5^1\phi_6^1$ $\xi_2 = \phi_1\phi_2\phi_3^1\phi_4^1\phi_6^1\phi_5^1$ $\xi_3 = \phi_1\phi_2\phi_3^2\phi_4^2\phi_5^2\phi_6^2$ $\xi_4 = \phi_1\phi_2\phi_3^2\phi_4^2\phi_6^2\phi_5^2$	$\frac{1}{8}$ $\frac{1}{8}$ $\frac{3}{8}$ $\frac{3}{8}$

The observations are therefore as follows:

End Users	*Observed Traces*	*Probability*
$\frac{2}{3}.u_1$	$obs(\xi_1) = abdfh$ $obs(\xi_2) = abdhf$ $obs(\xi_3) = acegk$ $obs(\xi_4) = acekg$	$\frac{1}{6}$ $\frac{1}{6}$ $\frac{1}{3}$ $\frac{1}{3}$
$\frac{1}{3}.u_2$	$obs(\xi_1) = abdfh$ $obs(\xi_2) = abdhf$ $obs(\xi_3) = acegk$ $obs(\xi_4) = acekg$	$\frac{1}{8}$ $\frac{1}{8}$ $\frac{3}{8}$ $\frac{3}{8}$

The entropy-based information loss due to the behaviour of u_1 is:

$$\mathcal{I}_{u_1}(T_{u_1}; O_{u_1}) = \mathcal{H}\left(\frac{1}{6}, \frac{1}{6}, \frac{1}{3}, \frac{1}{3}\right) = 1.9183\ . \tag{38}$$

The entropy-based information loss due to the behaviour of u_2 is:

$$\mathcal{I}_{u_2}(T_{u_2}; O_{u_2}) = \mathcal{H}\left(\frac{1}{8}, \frac{1}{8}, \frac{3}{8}, \frac{3}{8}\right) = 1.8113\ . \tag{39}$$

The channel capacity is therefore $C = \max_{u \in U}\ \mathcal{I}(T_u; O_u) = 1.9183$ yielding $\widehat{\pi}_{1.9183}$-opacity.

We now assume that if service provider 1 *breaks down then* service provider 2 *is used as a replacement. If the availability of* service provider 1 *is x, we obtain the following:*

End Users	*Observed Traces*	*Probability*
$\frac{2}{3}.u_1$	$obs(\xi_1) = abdfh$ $obs(\xi_2) = abdhf$ $obs(\xi_3) = acegk$ $obs(\xi_4) = acekg$	$\frac{1}{6} \cdot x$ $\frac{1}{6} \cdot x$ $\frac{1}{3} + \frac{1}{6} \cdot (1 - x)$ $\frac{1}{3} + \frac{1}{6} \cdot (1 - x)$
$\frac{1}{3}.u_2$	$obs(\xi_1) = abdfh$ $obs(\xi_2) = abdhf$ $obs(\xi_3) = acegk$ $obs(\xi_4) = acekg$	$\frac{1}{8} \cdot x$ $\frac{1}{8} \cdot x$ $\frac{3}{8} + \frac{1}{8} \cdot (1 - x)$ $\frac{3}{8} + \frac{1}{8} \cdot (1 - x)$

We therefore have:

$$\begin{aligned} \mathcal{I}_{u_1}(T_{u_1}; O_{u_1}) &= \mathcal{H}\left(\tfrac{x}{6}, \tfrac{x}{6}, \tfrac{3-x}{6}, \tfrac{3-x}{6}\right) \\ \mathcal{I}_{u_2}(T_{u_2}; O_{u_2}) &= \mathcal{H}\left(\tfrac{x}{8}, \tfrac{x}{8}, \tfrac{4-x}{8}, \tfrac{4-x}{8}\right)\ . \end{aligned} \tag{40}$$

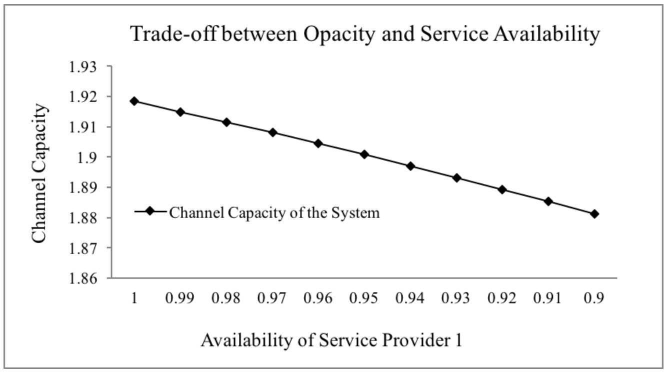

Fig. 3. The trade-off between service availability and opacity.

Figure 3 shows the channel capacity based opacity of the system against the availability of service provider 1*. Since in real life distributed systems the number of the parameters is high, in our study the probability of all possible runs of u_1 was fixed at $\frac{2}{3}$, and that of u_2 at $\frac{1}{3}$.*

The graph of Figure 3 clearly indicates that a system with high level of availability of service provider 1 *would suffer from low degree of opacity or, in other words, high information leakage.*

Looking at the above example, we can observe that a system with high level of availability may lead to high information leakage. Moreover, different security preserving mechanisms or policies might produce different observations. Given a computing system, it is therefore important to study the relations among the system security policies from the point view of the resulting degree of security. The following result suggests an ordering on such relations.

Theorem 1. *Let Pol_1 and Pol_2 be security policies for architectures with public and private clouds, which yield static observations functions obs_1 and obs_2, respectively. For a set of runs T, these observation functions define partitions, η_1 and η_2, of T into equivalence classes. Moreover, O_1 and O_2 are the observed runs under obs_1 and obs_2, respectively. Then:*

$$\eta_1 \sqsubseteq \eta_2 \implies \mathcal{I}(T;O_1) \geq \mathcal{I}(T;O_2) \ , \tag{41}$$

where $\eta_1 \sqsubseteq \eta_2$ means that for every $S_1 \in \eta_1$ there is $S_2 \in \eta_2$ such that $S_1 \subseteq S_2$.

Note: The above implies that if η_2 is coarser than η_1 then the information loss under η_2 is smaller than under η_1.

Proof: η_2 groups sets of runs of η_1, which adds ambiguity, and hence reduces mutual information between T and O. Therefore, the uncertainty measure of actual runs T given the condition O_2 is bigger than or equal to that of T given the condition O_1, i.e., $\mathcal{H}(T|O_2) \geq \mathcal{H}(T|O_1)$. Hence:

$$\begin{aligned}&\mathcal{I}(T;O_1) - \mathcal{I}(T;O_2)\\ &\quad = (\mathcal{H}(T) - \mathcal{H}(T|O_1)) - (\mathcal{H}(T) - \mathcal{H}(T|O_2))\\ &\quad = \mathcal{H}(T|O_2) - \mathcal{H}(T|O_1) \geq 0 \ .\end{aligned}$$

Thus $\eta_1 \sqsubseteq \eta_2$ implies $\mathcal{I}(T;O_1) \geq \mathcal{I}(T;O_2)$. □

7 Cost Model for Opacity and Security

In this section, we introduce a cost function, which needs to be optimized. The security policy is an architecture with public/private clouds, yielding an observation function, and a predicate that needs to remain opaque. The cost function is based on the assumption that there is a cost of security policy and a competing cost of opacity. The cost of the security policy is the financial cost of setting up and using a particular architecture. The cost of opacity is the financial cost of leaking confidential information that should remain hidden. This gives rise to the following cost function:

$$total\_cost = policy\_cost + opacity\_cost \tag{42}$$

The security policies plays a key role in determining the best design of the system. In addition, the security policies also influence the opacity of the system.

Example 6. *Consider Example 4 and the scenarios in Section 4.1. One can use the entropy-based opacity measurement to analyse the cost of the system, as it implies a degree of transparency of the communication channel between the users and the services in the system under different security preserving mechanisms.*

The two scenarios represent two security policies Pol_1 and Pol_2 in Section 4.1. Pol_1 indicates that clouds Y and X are private clouds. Messages communicated between the clouds X, Y and local network are visible, and messages inside the clouds are invisible. Pol_2 indicates that Y is public and X is private. Messages communicated between the clouds X, Y and the local network are visible, messages inside Y are visible, and messages inside X are invisible. From these decisions one can calculate $policy\_cost_1$ and $policy\_cost_2$.

By applying the measurement mechanism, we also have that the entropy-based opacity related to Pol_1 is $\overline{\pi}_1$, and the opacity related to Pol_2 is $\overline{\pi}_0$, see Example 4. By taking into account the specific financial cost of leaking information one can evaluate $opacity\_cost_1$ and $opacity\_cost_2$.

The resulting $total\_cost_1$ and $total\_cost_2$ can then be compared and the better option selected for deployment.

8 Conclusion

Federated cloud systems increase the reliability and reduce the cost of computational support. However, the large number of services and data involved creates security risks due to the dynamic movement of the entities between the clouds. A key role of information flow security is to ensure that information propagates throughout the execution environment without security violation; in particular, that no secure information is leaked to unauthorised subjects.

In this paper, a probabilistic information flow model is presented to analyse workflows deployed on federated cloud systems. Moreover, the notion of opacity is discussed as a security property in the analysis of systems. Following that, a threat model is proposed to analyze the flow sensitive security model based on the observations of users' behavioural patterns. Observational equivalence, entropy, and channel capacity are used to quantify opacity. As a result, trade-offs between opacity and service availability can be analysed. In addition, a cost model is presented to analyse the opacity and security policy of the system. The study presented in this paper can help service providers to allocate services and resources within federated cloud systems, and to make security related decisions.

As further research we envisage the development of effective verification techniques based on the results of this paper, e.g., using tools and algorithms from the Petri net field [27], [28]. Another direction for future research is to investigate — using verification tools — a range of realistic case studies in order to profile the usefulness of the different notions of opacity discussed in this paper and position them against other ways of measuring quantitative information flow, such as min-entropy leakage [35] and g-leakage [36], in the context of cloud computing.

Acknowledgement

We are very grateful to the anonymous reviewers for their detailed and constructive comments and suggestions.

References

[1] P. Watson, "A multi-level security model for partitioning workflows over federated clouds," *Journal of Cloud Computing*, vol. 1, no. 1, pp. 1 – 15, 2012.

[2] D. Bell and L. LaPadula, “Secure computer systems: Mathematical foundations,” MITRE Corporation, Tech. Rep., Mar. 1973.

[3] K. Knorr, “Multilevel security and information flow in Petri net workflows,” in *9th International Conference on Telecommunication Systems - Modeling and Analysis, Special Session on Security Aspects of Telecommunication Systems*, 2001.

[4] V. Varadharajan, “Hook-up property for information flow secure nets,” in *Computer Security Foundations Workshop IV, 1991. Proceedings*, 1991, pp. 154 – 175.

[5] K. Juszczyszyn, “Verifying enterprise’s mandatory access control policies with coloured Petri nets,” in *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on*, 2003, pp. 184 – 189.

[6] D. E. Bell, “Concerning ‘modeling’ of computer security,” in *Proceedings. 1988 IEEE Symposium on Security and Privacy*, 1988, pp. 8 – 13.

[7] H. Hiden, S. Woodman, and P. Watson, “A framework for dynamically generating predictive models of workflow execution,” in *Proceedings of WORKS 2013: 8th Workshop On Workflows in Support of Large-Scale Science, Held in conjunction with SC13, Denver, CO, USA, November 17, 2013*, 2013, pp. 77 – 87.

[8] J. Cala, H. Hiden, S. Woodman, and P. Watson, “Cloud computing for fast prediction of chemical activity,” *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1860 – 1869, 2013.

[9] S. Woodman, H. Hiden, and P. Watson, “Applications of provenance in performance prediction and data storage optimisation,” *Future Generation Computer Systems*, vol. 75, pp. 299 – 309, 2017.

[10] S. Sharif, P. Watson, J. Taheri, S. Nepal, and A. Y. Zomaya, “Privacy-aware scheduling saas in high performance computing environments,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1176 – 1188, 2017.

[11] J. A. Goguen and J. Meseguer, “Security policies and security models,” in *1982 IEEE Symposium on Security and Privacy*, April 1982, pp. 11 – 11.

[12] R. Denning, *Cryptography and data security*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1982.

[13] D. E. Denning, “A lattice model of secure information flow,” *Communications of the ACM*, vol. 19, pp. 236 – 243, May 1976.

[14] J. Landauer and T. Redmond, “A lattice of information,” in *Computer Security Foundations Workshop VI, 1993. Proceedings*, Jun 1993, pp. 65 – 70.

[15] J. K. Millen, “Covert channel capacity,” in *IEEE Symposium on Security and Privacy*, 1987, pp. 60 – 66.

[16] A. McIver and C. Morgan, *A probabilistic approach to information hiding*. New York: Springer New York, 2003, pp. 441 – 460.

[17] D. Volpano, C. Irvine, and G. Smith, “A sound type system for secure flow analysis,” *Journal of Computer Security*, vol. 4, pp. 167 – 187, January 1996.

[18] D. Clark, S. Hunt, and P. Malacaria, “A static analysis for quantifying information flow in a simple imperative language,” *Journal of Computer Security*, vol. 15, pp. 321 – 371, August 2007.

[19] J. Bryans, M. Koutny, and P. Y. A. Ryan, “Modelling opacity using Petri nets,” *Electronic Notes in Theoretical Computer Science*, vol. 121, pp. 101 – 115, February 2005.

[20] J. Bryans, M. Koutny, L. Mazaré, and P. Ryan, “Opacity generalised to transition systems,” in *Formal Aspects in Security and Trust*, ser. Lecture Notes in Computer Science, 2006, vol. 3866, pp. 81 – 95.

[21] ——, “Opacity generalised to transition systems,” *International Journal of Information Security*, vol. 7, pp. 421 – 435, 2008.

[22] M. S. Alvim, M. E. Andrés, and C. Palamidessi, “Probabilistic information flow,” in *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, 11-14 July 2010, Edinburgh, United Kingdom*, 2010, pp. 314 – 321.

[23] M. Andrés, C. Palamidessi, P. van Rossum, and G. Smith, “Computing the leakage of information-hiding systems,” in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. Lecture Notes in Computer Science, J. Esparza and R. Majumdar, Eds. Springer Berlin / Heidelberg, 2010, vol. 6015, pp. 373 – 389.

[24] M. E. Andrés, C. Palamidessi, P. van Rossum, and A. Sokolova, “Information hiding in probabilistic concurrent systems,” in *QEST 2010, Seventh International Conference on the Quantitative Evaluation of Systems, Williamsburg, Virginia, USA, 15-18 September 2010*, 2010, pp. 17 – 26.

[25] S. Hamadou, C. Palamidessi, and V. Sassone, “Quantifying leakage in the presence of unreliable sources of information,” *Journal of Computer and System Sciences*, vol. 88, pp. 27 – 52, 2017.

[26] S. Haar, “Types of asynchronous diagnosability and the reveals-relation in occurrence nets,” *IEEE Transactions on Automatic Control*, vol. 55, no. 10, pp. 2310 – 2320, 2010.

[27] V. Germanos, S. Haar, V. Khomenko, and S. Schwoon, “Diagnosability under weak fairness,” in *14th International Conference on Application of Concurrency to System Design, ACSD 2014, Tunis La Marsa, Tunisia, June 23-27, 2014*, 2014, pp. 132 – 141.

[28] ——, “Diagnosability under weak fairness,” *ACM Transactions on Embedded Computing Systems*, vol. 14, no. 4, pp. 69:1 – 69:19, 2015.

[29] S. Haar, S. Haddad, T. Melliti, and S. Schwoon, “Optimal constructions for active diagnosis,” *Journal of Computer and System Sciences*, vol. 83, no. 1, pp. 101 – 120, 2017.

[30] C. E. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal*, vol. 27, pp. 379 – 423, 1948.

[31] W. Zeng, M. Koutny, and P. Watson, “Verifying secure information flow in federated clouds,” in *IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom)*, 2014.

[32] W. Zeng, M. Koutny, P. Watson, and V. Germanos, “Formal verification of secure information flow in cloud computing,” *Journal of Information Security and Applications*, vol. 27-28, pp. 103 – 116, 2016.

[33] W. Zeng, M. Koutny, and P. Watson, “Opacity in internet of things with cloud computing (short paper),” in *8th IEEE International Conference on Service-Oriented Computing and Applications, SOCA 2015, Rome, Italy, October 19-21, 2015*, 2015, pp. 201 – 207.

[34] W. Zeng and V. Germanos, “Benefit and cost of cloud computing security,” *Harnessed Causality: Essays Dedicated to Maciej Koutny on the Occasion of His 60th Birthday*, pp. 143 – 150, 2018.

[35] G. Smith, “On the foundations of quantitative information flow,” in *Foundations of Software Science and Computational Structures, 12th International Conference, FOSSACS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*, 2009, pp. 288 – 302.

[36] M. S. Alvim, K. Chatzikokolakis, C. Palamidessi, and G. Smith, “Measuring information leakage using generalized gain functions,” in *25th IEEE Computer Security Foundations Symposium, CSF 2012, Cambridge, MA, USA, June 25-27, 2012*, 2012, pp. 265 – 279.

Wen Zeng is a VC2020 Lecturer in Cyber Security at School of Computer Science and Informatics, De Montfort University, United Kindom, and a guest member of staff at the School of Computing, Newcastle University. Recently, she has been working together with Maciej Koutny and Brian Randell on occurrence nets which can be used to detect service failure and malicious behavior in cloud computing systems. She has a strong research background in computing science and cyber security. In 2014, she was awarded her PhD on Quantitative Analysis of Distributed Systems from Newcastle University, working under the supervision of Maciej Koutny.

Maciej Koutny is a Professor in the School of Computing, Newcastle University, United Kingdom. In 1984 he received his PhD in Applied Mathematics from the Warsaw University of Technology, Poland, and after that joined the then Computing Laboratory of the University of Newcastle upon Tyne. His research interests centre on the theory of distributed and concurrent systems, including both theoretical aspects of their semantics and application of formal techniques to the modelling, synthesis and verification of such systems. In particular, he has been working on the development of formal models combining Petri nets and process algebras as well as on Petri net based behavioural models of membrane systems and reaction systems. He is an Adjunct Professor at McMaster University, Canada. In 2011 he held the Pascal Chair at Leiden University, The Netherlands. He has also been a Visiting Professor at several academic institutions.