A Flow Sensitive Security Model for Cloud Computing Systems

W. Zeng, C. Mu and M. Koutny

Abstract

Federated cloud systems increase the reliability and reduce the cost of computational support to an organization. The resulting combination of secure private clouds and less secure public clouds impacts on the security requirements of the system. To meet these security requirements, applications need to be located within different clouds, which strongly affects the information flow security of the entire system. In this paper, a flow sensitive security model for a federated cloud system is proposed, secure information flow in such a system is analyzed using coloured Petri nets, and opacity of cloud computing systems is introduced. In this study, the entities of a federated cloud system are assigned security levels of a given flow lattice. A transition system is used to describe the behavior of the system, and coloured Petri nets are used to analyze the correctness of the entire system. As a result, one can track the information flow. Moreover, one can analyze the impact of different resource allocation strategies, and the opacity of the system.

© 2013 Newcastle University.
Printed and published by Newcastle University,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.
Abstract

Federated cloud systems increase the reliability and reduce the cost of computational support to an organization. The resulting combination of secure private clouds and less secure public clouds impacts on the security requirements of the system. To meet these security requirements, applications need to be located within different clouds, which strongly affects the information flow security of the entire system. In this paper, a flow sensitive security model for a federated cloud system is proposed, secure information flow in such a system is analyzed using coloured Petri nets, and opacity of cloud computing systems is introduced. In this study, the entities of a federated cloud system are assigned security levels of a given flow lattice. A transition system is used to describe the behavior of the system, and coloured Petri nets are used to analyze the correctness of the entire system. As a result, one can track the information flow. Moreover, one can analyze the impact of different resource allocation strategies, and the opacity of the system.

About the authors

Wen Zeng is a Ph.D candidate in the School of Computing Science under the supervision of Professor Maciej Koutny. Her research interests centre on quantitative analysis of distributed system, including information security and privacy protection, information flow, opacity and probabilistic modelling.

Chunyan Mu received her PhD from King's College London, and is currently a Post-doc in LIAFA, Université Paris Diderot, Paris 7, Paris, France. Her research interests include: language-based security, programming languages, information flow security.

Maciej Koutny is a Professor in the School of Computing Science, Newcastle University. 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 and verification of such systems; in particular, model checking based on net unfoldings. He has also investigated non-interleaving semantics of priority systems, and the relationship between temporal logic and process algebras. Recently, he has been working on the development of a formal model combining Petri nets and process algebras as well as on Petri net based behavioural models of membrane systems.

Suggested keywords

OPACITY
COLOURED PETRI NETS
INFORMATION FLOW SECURITY
FEDERATED CLOUD SYSTEM
INFORMATION LATTICE
A Flow Sensitive Security Model for Cloud Computing Systems

Wen Zeng\(^1\), Chunyan Mu\(^2\), and Maciej Koutny\(^1\)

\(^1\) School of Computing Science, Newcastle University
Newcastle upon Tyne NE1 7RU, U.K.
wen.zeng.wz@gmail.com, maciej.koutny@ncl.ac.uk

\(^2\) LIAFA, Université Paris Diderot, Paris 7
Paris, France
chunyan.mu@liafa.univ-paris-diderot.fr

Abstract. Federated cloud systems increase the reliability and reduce the cost of computational support to an organization. The resulting combination of secure private clouds and less secure public clouds impacts on the security requirements of the system. To meet these security requirements, applications need to be located within different clouds, which strongly affects the information flow security of the entire system. In this paper, a flow sensitive security model for a federated cloud system is proposed, secure information flow in such a system is analyzed using coloured Petri nets, and opacity of cloud computing systems is introduced.

In this study, the entities of a federated cloud system are assigned security levels of a given flow lattice. A transition system is used to describe the behavior of the system, and coloured Petri nets are used to analyze the correctness of the entire system. As a result, one can track the information flow. Moreover, one can analyze the impact of different resource allocation strategies, and the opacity of the system.

Keywords: opacity, coloured Petri nets, information flow security, federated cloud system, information lattice

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 them in its large server clouds. However, large organizations may wish to keep sensitive information on their more restricted servers rather than in the public cloud. This has led to the introduction of federated cloud computing (FCC) in which both public and private cloud computing resources are used [17, 18].

A federated cloud is the deployment and management of multiple cloud computing services with the aim of matching business needs. Data, services, and
software are required to be allocated in different clouds for both security and business concerns. Although federated cloud systems (FCSs) can increase the reliability and reduce the cost of computational support to an organization, the large number of services and data on a cloud system creates security risks. As a result, it is very hard for an organization to track and control the information flow in the system. It is therefore necessary to develop a formal model describing the information flow security within an FCS, making the information and data traceable.

There exist different methods for addressing workflow security; for example, the flow-sensitive analysis of programs [16, 10, 15]. Using Petri nets to model the workflow, [12] applied Bell-LaPadula model to workflow security. However, the deployment of blocks within a workflow across a set of computational resources was not considered. The paper [18] proposed to partition workflows over a set of available clouds in such a way that security requirements are met. Such an approach is based on a multi-level security model that extends Bell-LaPadula to encompass cloud computing. [18] also indicated that workflow transformations are needed with data being communicated between clouds. However, the concurrency of the events or the execution of tasks in the system was not considered.

In this paper, we introduce a transition system representation to capture the information flow in a federated cloud system, and to use Coloured Petri Nets (CPNs) to analyze the security of information flow. Moreover, the opacity of cloud computing systems is discussed in order to help organizations in the analysis of the impact of different resource allocation strategies.

The paper is organized as follows. In Section 2, information lattices for secure information flow are introduced. In Section 3, a model for secure information flow analysis in FCSs is presented. A CPN model for the analysis of secure information flow is outlined in Section 4 using a case study. Finally, the opacity of cloud computing system is discussed in Section 5.

2 Information Lattices

In this section, we introduce security lattices for the components of a cloud system as well as for sets of individual clouds.

A lattice for security concerns [7, 8, 13] \(\mathcal{L} = (L, \leq)\) consists of a set \(L\) and a partial order relation \(\leq\) such that, for all \(l, l' \in L\), there exists a least upper bound \(l \sqcup l' \in L\), and a greatest lower bound \(l \sqcap l' \in L\). The lattice is complete if each subset \(L'\) of \(L\) has both a least upper bound \(\bigsqcup L'\) and a greatest lower bound \(\bigsqcap L'\).

Throughout the paper, we will assume that the origin of a federated cloud is a set \(\mathcal{C}\) of single deployment clouds. Moreover, \(\mathcal{S}\) will denote subjects (e.g. services, programs and processes), and \(\mathcal{O}\) will denote objects (e.g. resources and messages). Subjects and objects will jointly be referred to as entities, and their set will be denoted by \(\mathcal{E}\).

Following [13], at the centre of our approach is a complete security lattice \(\mathcal{L}_{\text{sec}} = (L_{\text{sec}}, \leq_{\text{sec}})\) with the ordering \(\leq_{\text{sec}}\) on security levels in \(L_{\text{sec}}\).
We will assign a security (or confidentiality) level \( l(e) \in L_{sec} \) to any entity \( e \in \mathcal{E} \) which will in practice be related to the degree of security of the contents of \( e \). Moreover, each cloud \( c \in \mathcal{C} \) will also be assigned a security level \( l(c) \in L_{sec} \). Intuitively, \( l(c) \) specifies the highest allowed security level of the entities located in \( c \).

3 System Model

Information flow security is concerned with the way in which secure information is allowed to flow through a computing system. Intuitively, the flow is considered secure if it adheres to a specified security policy. In this section, a formal model is introduced for capturing the information flow in federated cloud computing systems. The state transitions of the model can then be analyzed to verify that they satisfy conditions of a given security policy such as non-interference properties [9], Bell-Lapadula rules [3] for confidentiality considerations, Biba policies [4], and user-specified policies.

We now introduce a framework for security models of federated cloud computing systems based on guarded actions, each guard capturing security constraints introduced in order to ensure the security properties of interest.

Throughout this section, we assume that \( \mathcal{C} \) and \( \mathcal{E} \) are finite non-empty sets of respectively clouds and entities (or entity names), and \( L_{sec} \) is a security lattice as defined above. We also assume that there is a mapping \( l : \mathcal{C} \cup \mathcal{E} \to L_{sec} \) assigning security levels to individual clouds and entities.

In what follows, an entity can have different copies, and each of these copies can have a different security level and may reside in a different cloud. We further allow multiple copies of a single entity to be present in a single deployment cloud. As a result, in what follows, we will mean by a state any finite multiset \( s \) over the set \( \mathcal{E} \times \mathcal{C} \times L_{sec} \). Thus, for example, if \( s(e, c, 2) = 4 \) then we know that in the current state there are 4 copies of entity \( e \) with security level 2 residing in cloud \( c \). The elements of \( \mathcal{E} \times \mathcal{C} \times L_{sec} \) will be referred to as actual entities. We will say that an actual entity \((e, c, l)\) is present in state \( s \) if \( s(e, c, l) > 0 \).

**Definition 1 (fssm).** A flow-sensitive security model is a pair

\[
FSSM = (\mathcal{A}, s_{init}),
\]

where \( \mathcal{A} \) is a finite set of actions and \( s_{init} \) is an initial state. It is assumed that each action is a triple

\[
\phi = (in, out, \Sigma)
\]

such that the first two components

\[
in = (e_1@c_1, \ldots, e_k@c_k) \quad \text{and} \quad out = (e_{k+1}@c_{k+1}, \ldots, e_{k+m}@c_{k+m})
\]

are finite tuples of entity-cloud pairs and \( \Sigma \subseteq L_{sec}^{k+m} \) is a (security) guard.
Note that security guards can be provided in the form of a suitable predicate over \( k + m \) variables and suitable constants (such as security levels of given clouds).

**Definition 2 (single action execution).** An action \( \phi \) as in (2) is \( \sigma \)-enabled at state \( s \) if \( \sigma = (l_1, \ldots, l_{k+m}) \in \Sigma \) is a tuple of security levels such that

\[
\phi_{\sigma}^{in} = \{(e_1, c_1, l_1), \ldots, (e_k, c_k, l_k)\} \leq s.
\]

Such an action can then be \( \sigma \)-executed leading to a new state \( s' \)

\[
s' = s - \phi_{\sigma}^{in} + \phi_{\sigma}^{out},
\]

where \( \phi_{\sigma}^{out} = \{(e_{k+1}, c_{k+1}, l_{k+1}), \ldots, (e_{k+m}, c_{k+m}, l_{k+m})\} \). We denote this by \( s \xrightarrow{\phi_{\sigma}} s' \).

The last definition captures the enabling and execution of a single action. The next definition lifts this to any group of actions executed simultaneously.

**Definition 3 (multiset action execution).** Let \( \Phi = \{\phi_1 : \sigma_1, \ldots, \phi_n : \sigma_n\} \) be a multiset such that each \( \phi_i \) is an action \( \sigma_i \)-enabled at state \( s \) and, moreover,

\[
\Phi^{in} = (\phi_1)^{in}_{\sigma_1} + \ldots + (\phi_n)^{in}_{\sigma_n} \leq s.
\]

Then \( \Phi \) can then be executed leading to a new state

\[
s' = s - \Phi^{in} + \Phi^{out},
\]

where \( \Phi^{out} = (\phi_1)^{out}_{\sigma_1} + \ldots + (\phi_n)^{out}_{\sigma_n} \). We denote this by \( s \xrightarrow{\Phi} s' \).

**Definition 4 (reachable states).** The set of reachable states of the flow-sensitive security model as in (1) is the minimal set of states \( RS \) containing \( s_{init} \) and such that if \( s \in RS \) and \( s \xrightarrow{\Phi} s' \), for some \( \Phi \), then \( s' \in RS \).

We have defined general notions related to the syntax and operational semantics of a flow-sensitive security model. It is then straightforward to capture the basic notion of security across the federated cloud.

**Definition 5.** A state \( s \) is secure if, for every actual entity \((e, c, l)\) present in \( s \), \( l \leq_{sec} l(c) \). A flow-sensitive security model as in (1) is secure if all its reachable states are secure.

Intuitively, a state is secure if all copies of entities present in the state reside in clouds without causing security violation. One can formulate a general security policy guaranteeing the security of a flow-sensitive security model. Such a policy is formulated by placing a suitable condition on the security guards present in the model.

---

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

4 Note that \( \leq \) denotes multiset inclusion.

5 Note that ‘−’ and ‘+’ denote multiset subtraction and addition, respectively.
Theorem 1. Let FSSM be a flow-sensitive security model as in (1) such that,
for every action $\phi$ as in (2), $\{c_{k+1}, \ldots, c_{k+m}\} \subseteq \{c_1, \ldots, c_k\}$ and if $(l_1, \ldots, l_{k+m}) \in \Sigma$ then, for every $i = k + 1, \ldots, k + m$:
\[
\prod_{p > k | c_p = c_i} l_p \leq \text{sec} \prod_{r \leq k | c_r = c_i} l_r.
\]

Then FSSM is secure provided that its initial state is secure.

The above result can only be applied in specific cases; in general, we need
to verify that a given system specification yields a secure system. In the next
section, we will outline how coloured Petri nets can be used to provide a suitable
analytical tool.

4 Coloured Petri Net Model

Coloured Petri Nets (CPNs) can be used to model concurrent systems, includ-
ing information flow in FCSs. In particular, Petri net theory provides powerful
analysis techniques which can be used to verify the correctness of workflow proce-
dures [1, 2]. Transitions in Petri nets can be interpreted as occurrences of events
or executions of tasks in a system. In our representation of FCSs in CPNs, the
relations between events are explicit, and the representation of the system can
alleviate the state explosion problem [6].

CPNs allow one to model multi-type cases in a process specification [11],
which can model different entities of an FCS. Each token can carry complex
information and/or data. Arc expressions and multiple exits can be used to
model the various workflow logics. Moreover, guard functions associated with
transitions can be used to specify security-related conditions.

![Fig. 1. The medical research application example from [18].](image)

4.1 Case study

As a case study adapted to our purposes, we will use a simplified version of the
federated cloud system of [18]. It is a medical research application in which data
from a set of patients’ heart rate monitors is analyzed, as illustrated in Figure 1.

Informally, the process can be described as follows:
The input data \((d_0)\) is a file with a header identifying the patient (name and patient number), followed by a set of heart rate data recorded over a period of time;

- A service \((s_1)\) strips off the header, leaving only the heart rate data \((d_2)\);
- A second service \((s_3)\) analyzes the heart rate data, and produces results \((d_4)\).

Analyzing the heart rate data \((s_3)\) 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 use two clouds, one public cloud \(c_0\) and one private cloud \(c_1\). The proposed workflow operates on sensitive medical data processed on the private cloud, and anonymized data that can be deployed on the public cloud.

Figure 2 is a workflow which is derived from Figure 1, with the security settings shown in Table 1, where \{\(c_0, c_1\}\) are the clouds in the system, \{\(s_1, s_3\}\) are the services, and \{\(d_0, d_2, d_4\}\) are the data.

<table>
<thead>
<tr>
<th>Clouds</th>
<th>Security level</th>
<th>Services</th>
<th>Security level</th>
<th>Data</th>
<th>Security level</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c_0)</td>
<td>0</td>
<td>(s_1)</td>
<td>1</td>
<td>(d_0)</td>
<td>1</td>
</tr>
<tr>
<td>(c_1)</td>
<td>1</td>
<td>(s_3)</td>
<td>0</td>
<td>(d_2)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(d_4)</td>
<td>0</td>
</tr>
</tbody>
</table>

After determining valid mappings of services and data to clouds based on the Bell-Lapadula rules, the workflow in Figure 2 leads to six valid partitionings.

Fig. 3. A valid workflow partitioning for the FCS of [18].
The partitioning chosen for this case study is shown in Figure 3: \( \{ s_1, d_0, d_2 \} \) are services and data deployed on cloud \( c_1 \) whose security level is 1, while \( \{ s_3, d_2, d_4 \} \) are deployed on cloud \( c_0 \) whose security level is 0.

### 4.2 CPN model

Based on the FCS model outlined above, we built a CPN model capturing the information flow of the FCS in the case study, shown in Figure 4.

The model provides an abstract view of CPN model of the chosen workflow partitioning of the FCS. It has two places \( \text{cloud}_1 \) and \( \text{cloud}_0 \), which represent the two clouds, and two transitions, \( t_1 \) and \( t_2 \). Places store the entities of the system, which have the type \( \text{entity} \) as the colour set (describing the data are needed to be processed in the system). The declarations of the colour sets in Figure 4 tell us that that \( \text{entity} \) is a union type, which corresponds to a datatype in Standard ML, with the values being are either data \( (\text{data}) \) or services \( (\text{serv}) \). \( \text{data} \) and \( \text{serv} \) have an associated product colour sets which allows one to distinguish each individual entity. Thus, the type \( \text{entity} \) contains the values \( \{ \text{data}(d(0), 1, 1), \text{serv}(s(1), 1, 1), \ldots \} \). Each token has a colour which has

![Fig. 4. CPN model (including the colour sets) for a federated cloud system.](image-url)
three fields. The first field is an element of dName or sName, and thus it is
d0 or d1 or d2 or d3 or d4 or s1 or s2 or s3, which specifies the name of the
data/services in the system. The second element is an integer which specifies
the security level of the cloud where the service/data is located. The third ele-
ment is also an integer, specifying the security level of the service/data. Each of
the two transitions, t1 and t2, represents a move from one state to the next.

5 Opacity in Cloud Computing Systems

Observing behaviour patterns of users can lead to leakages of secure information.
Information sharing means that the behaviour of one cloud user may appear vis-
able to other cloud users or adversaries, and observations of such behaviours can
potentially help adversaries to build covert channels. Opacity is a uniform ap-
proach for describing security properties expressed as predicates [5]. A predicate
is opaque if an observer of the system is unable to determine the truth of the
predicate in a given run of the system. In this section, we will discuss one of
the versions of opacity in the context of workflows executed on federated cloud
computing systems.

Let FSSM = (A, sinit) be a flow-sensitive security model as in (1). A run of
FSSM is a finite sequence
\[ \xi = \Phi_1 \ldots \Phi_n \quad (n \geq 0) \] (3)
such that there are states \( s_{\text{init}} = s_1, \ldots, s_{n+1} \) satisfying \( s_i \xrightarrow{\Phi_i} s_{i+1} \) for \( i = 1, \ldots, n \). The set of all runs of FSSM will be denoted by \( \text{RUN}(\text{FSSM}) \).

To model the different capabilities for observing the system modelled by
FSSM one can use observation functions:
\[ \text{obs} : \text{RUN}(\text{FSSM}) \to \text{Obs}^* \]
where Obs be a set of observables. In what follows, we consider a state observation
function obs for which there is a map \( \text{obs}' \) associating \( \text{obs}'(\Phi) \in \text{Obs} \cup \{ \epsilon \} \) with every \( \Phi \) as in Definition 3, in such a way that
\[ \text{obs}(\xi) = \text{obs}'(\Phi_1) \ldots \text{obs}'(\Phi_n) \]
for every run \( \xi = \Phi_1 \ldots \Phi_n \) in \( \text{RUN}(\text{FSSM}) \).

Given the observation function obs, we are now interested in whether an
observer can establish a property \( \gamma \) (a predicate over system runs) for a run of
FSSM having only access to the result of the observation function. As one
can identify \( \gamma \) 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 \( \gamma \subseteq \text{RUN}(\text{FSSM}) \) can be deduced by the observer on the basis of an
observed execution of the system. Moreover, we are interested in the final opa-
cy predicate \( \gamma_Z \), defined as the set of all the runs \( \xi \) as in (3) satisfying \( s_n \in Z \), for
some set of states \( Z \) [5]. Intuitively, this means that we are interested in finding
out whether an observed run of the system represented by \textit{FSSM} ended in one of secret (or sensitive) states belonging to \( Z \). Note that we are not interested in establishing whether the underlying run does not belong to \( \gamma \); to do this, we would consider the property \( \bar{\gamma} = \operatorname{RUN}(\textit{FSSM}) \setminus \gamma \).

We then say that \( \gamma \) is \textit{opaque} w.r.t. \( \text{obs} \) if, for every run \( \xi \in \gamma \), there is another run \( \xi' \in \bar{\gamma} \) such that \( \text{obs}(\xi) = \text{obs}(\xi') \). In other words, if all runs in \( \gamma \) are covered by runs in \( \bar{\gamma} \):

\[
\text{obs}(\gamma) \subseteq \text{obs}(\bar{\gamma}).
\]

5.1 Case study

The scenario of this case study involves three clouds (\( W, X, \) and \( Y \)) and a number of processes, which all together form an \textit{e:shop} application. Cloud \( X \) hosts a web portal \textit{e:win}; Cloud \( W \) hosts two providers, \textit{e:prov1} and \textit{e:prov2}; and cloud \( Y \) hosts a payment handling site, \textit{e:pay}, and an internet bank, \textit{e:bank}. The functionality of the \textit{e:shop} is built around the transmission of information between the various predefined process participants, such as the providers and bank.

![Diagram](image)

\textbf{Fig. 5.} Information flow in a cloud based \textit{e:shop}.

Figure 5 shows the basic structure of the execution scenario for the \textit{e:shop}. The role of each of the processes is as follows:

- \textit{client} tries to buy a product online, accessing to cloud based retail applications through a web browser.
- \textit{e:win} is a web portal which provides a platform for trading.
- \textit{e:prov1} and \textit{e:prov2} supply products traded by \textit{e:shop}.
- \textit{e:pay} is an agent through which \textit{client} can make online payment and transfer money between bank accounts.
- \textit{e:bank} can handle payments and deposits through the \textit{e:pay} agent.
The generic behaviour of $e:shop$ is shown in Table 2. It starts with a purchase request sent from $client$ to $e:win$. The request is forwarded to the product providers, $e:prov1$ and $e:prov2$, who reply with the relevant product information. $e:win$ forwards the received information to $client$, who selects one of the two products and sends the decision back to $e:win$. Then $e:win$ forwards the decision to $e:pay$ which contacts $client$ asking for payment details. After receiving the payment information, $e:pay$ contacts $e:bank$ to carry out the payment. Then $e:pay$ sends an invoice to $client$ and informs $e:win$. Finally, $e:win$ contacts the selected provider to trigger the shipment of the product to $client$. Moreover, the selected provider is not allowed to reveal their identity to an observer.

**Table 2.** The sequence of interactions between the components of the $e:shop$ system.

<table>
<thead>
<tr>
<th>Entities (subject)</th>
<th>Actions</th>
<th>Sender</th>
<th>Receiver (object)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>client’s request</td>
<td>$\phi_1$</td>
<td>$client$</td>
</tr>
<tr>
<td>2</td>
<td>forward client’s request</td>
<td>$\phi_2$</td>
<td>$e:win$</td>
</tr>
<tr>
<td>3</td>
<td>product’s information</td>
<td>$\phi_3$</td>
<td>$e:prov$ ($i=1,2$)</td>
</tr>
<tr>
<td>4</td>
<td>forward product’s information</td>
<td>$\phi_4$</td>
<td>$e:win$</td>
</tr>
<tr>
<td>5</td>
<td>choose products</td>
<td>$\phi_5$</td>
<td>$client$</td>
</tr>
<tr>
<td>6</td>
<td>forward the client to the payment</td>
<td>$\phi_6$</td>
<td>$e:win$</td>
</tr>
<tr>
<td>7</td>
<td>client’s information request</td>
<td>$\phi_7$</td>
<td>$e:pay$</td>
</tr>
<tr>
<td>8</td>
<td>make payment</td>
<td>$\phi_8$</td>
<td>$client$</td>
</tr>
<tr>
<td>9</td>
<td>contact bank</td>
<td>$\phi_9$</td>
<td>$e:pay$</td>
</tr>
<tr>
<td>10</td>
<td>make payment</td>
<td>$\phi_{10}$</td>
<td>$e:bank$</td>
</tr>
<tr>
<td>11</td>
<td>send invoices</td>
<td>$\phi_{11}$</td>
<td>$e:pay$</td>
</tr>
<tr>
<td>12</td>
<td>inform of the payment</td>
<td>$\phi_{12}$</td>
<td>$e:pay$</td>
</tr>
<tr>
<td>13</td>
<td>inform of the payment</td>
<td>$\phi_{13}$</td>
<td>$e:win$</td>
</tr>
<tr>
<td>14</td>
<td>ship product</td>
<td>$\phi_{14}$</td>
<td>$e:prov$ ($i=1,2$)</td>
</tr>
</tbody>
</table>

We also assume that no provider is discriminated against. Moreover, messages communicated between the clouds $X$, $W$ and $Y$ are invisible, and other messages are visible. However, 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 the following (static) observation function:

$$
\begin{align*}
\text{obs}'(\phi_1) &= a \\
\text{obs}'(\phi_2) &= \epsilon \\
\text{obs}'(\phi_3) &= \epsilon \\
\text{obs}'(\phi_4) &= \epsilon \\
\text{obs}'(\phi_5) &= \epsilon \\
\text{obs}'(\phi_6) &= \epsilon \\
\text{obs}'(\phi_7) &= \epsilon \\
\text{obs}'(\phi_9) &= \epsilon \\
\text{obs}'(\phi_{10}) &= \epsilon \\
\text{obs}'(\phi_{11}) &= \epsilon \\
\text{obs}'(\phi_{12}) &= \epsilon \\
\text{obs}'(\phi_{13}) &= \epsilon \\
\text{obs}'(\phi_{14}) &= \epsilon
\end{align*}
$$

Using opacity, we may show that visible interaction do not reveal the identity of the provider supplying the goods. To see this, we consider a property $\gamma$.
consisting of all execution scenarios where the first provider supplied the goods, i.e., executions of the following form:

\[ \xi_1 = \Phi_1 \Phi_2 \Phi_3 \Phi_4 \Phi_5 \Phi_6 \Phi_7 \Phi_8 \Phi_9 \Phi_{10} \Phi_{11} \Phi_{12} \Phi_{13} \Phi_{14} \]

\[ \xi_2 = \Phi_1 \Phi_2 \Phi_3 \Phi_4 \Phi_5 \Phi_6 \Phi_7 \Phi_8 \Phi_9 \Phi_{10} \Phi_{11} \Phi_{12} \Phi_{13} \Phi_{14} \]

\[ \xi_3 = \Phi_1 \Phi_2 \Phi_3 \Phi_4 \Phi_5 \Phi_6 \Phi_7 \Phi_8 \Phi_9 \Phi_{10} \Phi_{11} \Phi_{12} \Phi_{13} \Phi_{14} \]

\[ \xi_4 = \Phi_1 \Phi_2 \Phi_3 \Phi_4 \Phi_5 \Phi_6 \Phi_7 \Phi_8 \Phi_9 \Phi_{10} \Phi_{11} \Phi_{12} \Phi_{13} \Phi_{14} \]

The set of observations they generate is given by \( \text{obs}(\gamma) = \{abacdexe}\). We then note that \( \tilde{\gamma} \) comprises, among others, executions of the following kind:

\[ \xi_1 = \Phi_1 \Phi_2 \Phi_3 \Phi_4 \Phi_5 \Phi_6 \Phi_7 \Phi_8 \Phi_9 \Phi_{10} \Phi_{11} \Phi_{12} \Phi_{13} \Phi_{14} \]

\[ \xi_2 = \Phi_1 \Phi_2 \Phi_3 \Phi_4 \Phi_5 \Phi_6 \Phi_7 \Phi_8 \Phi_9 \Phi_{10} \Phi_{11} \Phi_{12} \Phi_{13} \Phi_{14} \]

\[ \xi_3 = \Phi_1 \Phi_2 \Phi_3 \Phi_4 \Phi_5 \Phi_6 \Phi_7 \Phi_8 \Phi_9 \Phi_{10} \Phi_{11} \Phi_{12} \Phi_{13} \Phi_{14} \]

\[ \xi_4 = \Phi_1 \Phi_2 \Phi_3 \Phi_4 \Phi_5 \Phi_6 \Phi_7 \Phi_8 \Phi_9 \Phi_{10} \Phi_{11} \Phi_{12} \Phi_{13} \Phi_{14} \]

Hence \( \text{obs}(\gamma) \subseteq \text{obs}(\tilde{\gamma}) \), and so \( \gamma \) is an opaque property. As a result, it is never possible to say for sure that it was the first provider who supplied the goods. Since the argument is completely symmetric, we can conclude that the identity of providers is kept secret.

6 Conclusions

In this paper, a flow-sensitive security model is presented to analyze the information flow in the federated cloud system. The entities of the cloud system are typed into different security levels which form a security lattice ordered by the security levels, each cloud is also mapped into a security level in the security lattice to specify the confidential level of the cloud. Coloured Petri nets model is built to verify the correctness of the secure information flow. Opacity as a promising technique for unifying security properties is introduced to cloud computing systems. This study can help to track and control the secure information flow in federated cloud system, and it also can be used to analyze the impact of different resources allocation strategies.

References

2. W. van der Aalst: Verification of workflow nets. ICATPN’97 (1997)