Towards Security-based Formation of Cloud Federations: A Game Theoretical Approach

Talal Halabi, Student Member, IEEE, and Martine Bellaiche

Abstract—Cloud federations allow Cloud Service Providers (CSPs) to deliver more efficient service performance by interconnecting their Cloud environments and sharing their resources. However, the security of the federated service could be compromised if the resources are shared with relatively insecure CSPs, and violations of the Security Service Level Agreement (Security-SLA) might occur. In this paper, we propose a Cloud federation formation model that considers the security level of CSPs. We start by applying the Goal-Question-Metric (GQM) method to develop a set of parameters that quantitatively describes the Security-SLA in the Cloud, and use it to evaluate the security levels of the CSPs and formed federations with respect to a defined Security-SLA baseline, while taking into account CSPs' customers' security satisfaction. Then, we model the Cloud federation formation process as a hedonic coalitional game with a preference relation that is based on the security level and reputation of CSPs. We propose a federation formation algorithm that enables CSPs to join a federation while minimizing their loss in security, and refrain from forming relatively insecure federations. Experimental results show that our model helps maintaining higher levels of security in the formed federations and reducing the rate and severity of Security-SLA violations.

Index Terms—Cloud Computing, security evaluation, Security-SLA, Cloud federation, coalition formation, game theory.

1. Introduction

The adoption of Cloud Computing technology is presently on the rise, and the Cloud market is expecting to keep its rapid growing pace in the next few years [1]. By transferring their businesses to the Cloud, customers benefit from many interesting features such as scalability, resilience, high performance, on-demand and Pay-Per-Use service model. However, outsourcing services to a third party adds a new level of risk due to loss of control, and introduces many security threats such as data breaches, data loss, and denial of service [2], which makes security an essential driving factor of the Cloud market today. Customers expect from the Cloud Service Providers (CSPs) to maintain the security and availability of their data and services, and demonstrate compliance with current security standards. Evaluating the security of CSPs is a tough task, due to the difficulty in fully quantifying it, but it will permit to speed up the Cloud adoption process by providing sufficient and transparent information about the security of the offered services to customers and facilitating their comparison during the decision making process. The work on developing the adequate terms and policies that will form the future Cloud Security Service Level Agreement (Security-SLA) which will govern security management between providers and customers is presently very active. This agreement will help in guarantying the rights of each party and will require full commitment from both sides to avoid security violations and what entail in terms of financial and technical penalties.

The Cloud Computing paradigm also introduces important challenges for CSPs, such as performance guarantee, resource limitation, disaster-recovery planning, regional distribution of workloads, and legal issues. To address these problems, the concept of Cloud federation was born. It allows a CSP to flexibly and transparently outsource a portion of its users' requests to other independent CSPs, especially when the limitation of available resources on the CSP's side can't cope with the dynamic nature of the Cloud workload and the variability in users' requests for data and computing-intensive applications. This federation between CSPs can occur at different service delivery models: Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS), and along two different dimensions: horizontal, which takes place at matching layers of the Cloud Stack, and vertical, which spans multiple layers in order to service the additional requests on one specific layer through delegation [3]. By providing larger amounts of resources, federated Cloud infrastructures maintain higher performance and Quality of Service (QoS) levels, and improve cost-effectiveness and energy efficiency [3]. In addition, their objectives can also involve enhancing resilience against failures or unexpected situations, and redundancy implementation by replicating computations and data among CSPs to increase security or availability [4].

1.1. Problem Definition

In order to form a Cloud federation and allow CSPs to find their potential federation candidates, the state-of-the-art has proposed several models, principally based on factors such as profit maximization, trustworthiness, and QoS parameters. Security was ignored in the Cloud federation formation process, mainly due to the difficulty of its evaluation. For instance, joining a federation of trustworthy CSPs could ensure service reliability and maintain or enhance the required QoS level. Similarly, entering a federation that maximizes profit and reduces cost could be very beneficial to CSPs, but both approaches do not guarantee service safety. Forming a federation with insecure CSPs could increase security risk and hurt the reputation of the federation members. For example, if a CSP delegates its workload to another CSP with a security level that is lower than its own, it might risk to not satisfy all its clients’ security requirements and end up violating some of the terms in the Security-SLA. In addition, this kind of federation could be dangerous to a CSP since its federated VMs still can communicate with the VMs

The authors are with the Department of Computer and Software Engineering, École Polytechnique de Montréal, Montreal, Quebec, Canada. Email: {talal.halabi, martine.bellaiche}@polymtl.ca
that exist on its own infrastructure, which permits to an
attack or any malicious activity to propagate through the
infrastructures of the federation members.

Therefore, to guarantee security satisfaction, limit the
rate of security violations, reduce the cost of applied
penalties, and protect the Cloud infrastructures from the
risk of avoidable security threats, the level of security
services provided by the federation members should be
efficiently evaluated before the formation of federations.
Such services include identity and access management,
data and application security, encryption and key manage-
ment, virtualization security, and incident response plans.
Since other criteria such as pricing and QoS (i.e., response
time and availability) are also important for the formation
of Cloud federations, which is normally performed with
the objective of reducing deployment costs or increasing
the level of performance of large applications, these crite-
ria could eventually be combined along with the security
criterion to provide more efficient and secure services.
In this paper, we tackle the Cloud federation formation
problem within a security context. Our focus is on the IaaS
model and the horizontal federation between independent
Cloud infrastructures, where resources are shared between
the CSPs in the form of Virtual Machine (VM) instances
as illustrated in Fig. 1. This form of resource federation
is the basis of the other federation models, and performing it
securely will imply the protection of all kinds of federated
services. The security of the IaaS model is explicitly high-
lighted since the other service delivery models are built
on top of it and any security breach on the IaaS layer will
result in security compromise on the other layers.

1.2. Contribution

In this paper, we discuss the Cloud federation formation
problem while considering the security levels of CSPs.
We design a security-based hedonic coalitional game
that models the Cloud federation formation process, and
which goal is to increase the number of secure federations
by minimizing the number of insecure members within.
The game reduces the loss in the security level of CSPs
caused by moving from a non-cooperative state and join-
ing a federation, and tends to maintain a stable rate of
secure federations. Our contributions are stated as follows:

• First, we make use of the Goal-Question-Metric
  (GQM) [5] method to develop a set of quantitative
  Security-SLA parameters that evaluate the level of
  deployed security services on the different layers of
  the Cloud architecture.

• Second, we propose a method to evaluate and com-
  pute the security levels provided by CSPs’ Security-
  SLAs, and the security levels of formed federations
  relatively to a defined security baseline, while con-
  sidering CSPs’ reputations.

• Third, taking into account the computed security
  levels, we propose a Cloud federation formation al-
  gorithm based on a hedonic coalitional game with a
  security preference relation that satisfies the stability
  property [6], that is, none of the CSPs has incentive
to leave its current federation to join another one
  that is relatively more secure.

The properties of the security-based Cloud federation
formation game are then analyzed, and a set of experi-
ments is performed to study the efficiency of the proposed
algorithm. Results show that the model helps in separat-

1.3. Paper Organization

The remainder of the paper is structured as follows.
Section 2 discusses the literature review related to Cloud
security evaluation and federation formation. Section 3
gives an overview of the proposed security-based Cloud
federation formation framework. Section 4 addresses the
development and quantification of the Cloud Security-
SLA and the evaluation of security of CSPs and feder-
ations. Section 5 describes the security-based federation
formation game model that we propose. In Section 6,
experimental results are presented and analyzed. Finally,
section 7 concludes the paper.

2. Literature Review

Current security standards are helping CSPs to some
extent in implementing and evaluating their security sys-
tems. However, more work is still required to standardize
the evaluation of security in Cloud Computing, which
on one hand, will facilitate the comparison between the
different available Cloud services and the customer’s
decision making process, and on the other hand, will
create a competitive market where CSPs are encouraged
to deploy more secure and transparent services, speeding up
the Cloud adoption movement. The National Institute of
Science and Technology (NIST) [7] and the Cloud Security
Alliance (CSA) [9] are investing a lot of efforts in this
area through different projects like the Cloud Computing
Service Metrics Description [8] and the Cloud Controls
Matrix (CCM) within the Governance, Risk Management
and Compliance (GRC) Stack initiated by the CSA [9].
However, these projects still lack the security quantifica-
tion aspect. The SPECS project [10] is also an example of
such effort: Its main objective is to develop a framework
that offers Security-as-a-Service in the Cloud based on
the security parameters specified in the SLA, and provide
the necessary techniques to manage its life cycle. In our
previous work [11], we tried to quantify the evaluation
of security of CSPs and developed a potential set of
quantitative security metrics that can be used to evaluate
their deployed security services. One of our contributions
in this paper over our previous work is the evaluation
of security according to a well-defined baseline instead of
evaluating the security level of CSPs with respect to each
other. We review here some of the work on Cloud security
evaluation and the formation of Cloud federations.

Da Silva et al. [12] described a hierarchy of security met-
crics which they derived using the GQM methodology to
characterize security in the Cloud. The hierarchy is based
on categorizing the metrics according to their objectives
that permits the formation of multi-cloud communities of
trustworthy and reliable CSPs. In [24], Abdel Wahab et
al. also proposed a trust-based hedonic coalitional game
mechanism using a trust-based cooperative game theory
approach. In [23], Hassan et al. proposed a federation formation
tion of Cloud federations from a trustworthiness perspec-
tive. However, his model did not cover all critical
security aspects and services in the Cloud and did not lead to a possible quantitative evaluation methodology.
Na and Huh [14] developed a Cloud service selection
model based on the evaluation of Security-SLA. They
considered five of the nine most notorious security threats
to Cloud Computing [2] which they judged critical from
the user’s perspective: data breaches, data loss, account hijacking, insecure APIs, and malicious insiders, and mostly
focused on computing their subjective weights using the
Analytic Network Process (ANP) method. Zhengwei et
al. [15] proposed a quantifiable system of Cloud-oriented
Security-SLA indicators also using the AHP method. They divided their metrics according to the benefit and
cost types and used a nearness calculation strategy to
relatively evaluate the service level by computing the normal-
ized weighted distance from the real service level to the
best and worst service levels. Taha et al. [16] used the
Analytic Hierarchy Process (AHP) technique to evaluate
Cloud security based on the Security-SLA provided by the
Consensus Assessments Initiative Questionnaire (CAIQ)
of the CSA [9]. In [17], the same authors performed a quantitative assessment of a more elaborated Security-
SLA to evaluate the security levels of CSPs according to
customers’ requirements. However, their approaches
did not address the quantification and measurement of
security of CSPs with respect to a standard security
baseline, independently from customers’ security require-
mements, which is the approach we follow in this paper.
Finally, Di Vimercati et al. [18] modeled users’ require-
ments in a Cloud environment and used them to rank the
different Cloud plans according to satisfaction and
preference. However, their model did not discuss about
users’ security requirements in particular.

The concept of Cloud federations is still immature.
Many researchers have addressed the subject of Cloud
federation formation from a profit generation perspective.
For instance, Li et al. [19] proposed an algorithm for
VMs trading in a Cloud federation using an auction-
based scheduling mechanism that maximizes the profit of
the federation members. Samaan [20] designed a resource
sharing strategy in a Cloud federation that increases the
revenue based on game theory. Mashayekhy et al. [21]
introduced a Cloud federation formation game that allows
CSPs to maximize their profit. Finally, Guazzzone et al. [22]
used the cooperative game theory to develop an algorithm
that allows the formation of federations while maximizing the
profit of CSPs and reducing the energy cost.

Other research have addressed the problem of forma-
tion of Cloud federations from a trustworthiness perspec-
tive. In [23], Hassan et al. proposed a federation formation
mechanism using a trust-based cooperative game theory
that allows CSPs to maximize their profit and minimize the SLA penalty cost on QoS by joining federations of
trustworthy and reliable CSPs. In [24], Abdel Wahab et
al. also proposed a trust-based hedonic coalitional game
that permits the formation of multi-cloud communities of

trustworthy services. However, none of these approaches
have considered the security factor when addressing the
subject of federation formation, and which could have
serious consequences on the security of the formed federa-
tions. Evaluating the security level of CSPs and taking it
into account while forming the federations is the research
gap which we are trying to fill with our work in this paper.

3. The Proposed Security-Based Federation Formation Framework

The proposed framework for security-based Cloud fed-
eration formation involves two stages and is depicted in
Fig. 2. The first stage addresses the evaluation of security
in the Cloud and consists on, first, developing a Security-
SLA for the Cloud, that describes in quantifiable terms the
security provided by CSPs and measures its performance
and cost, and second, evaluating the security level of CSPs
and the possible federations that could form between
them. This evaluation takes into account the reputation
of CSPs with respect to security and which is computed
based on their customers’ satisfaction, and the amount of
resources they intend to share with the federation.

The second stage consists on applying a federation
formation algorithm that we propose based on a hedonic
coalitional game which takes into consideration a security
preference relation that we define. The security level com-
puted during the first stage will be fed to the formation
algorithm which will generate a set of Cloud federations.
In the proposed framework, the Cloud broker acts as an
intermediate party between the customers and providers.
The broker stores the information related to the Security-
SLA offered by each CSP along with the reputation values
of CSPs which are regularly updated based on the new
interactions with their customers. During the federation
formation game, the CSPs interact with the Cloud broker
in order to obtain the necessary information about each
other’s security levels and reputations. The framework is
discussed in detail in the following sections.


Cloud security evaluation creates many challenges, such
as the necessity to consider all security aspects in the
Cloud, the need to cover all the layers of the Cloud
architecture and what could entail of threats and incidents,
the difficulty in quantifying the security level of a Cloud
infrastructure, and the requirement for standardization
to establish a fair and reasonable evaluation of CSP’s security
offers. In this section, we propose a model of the Cloud
Security-SLA and a method to evaluate the security level
of CSPs based on their offered Security-SLAs.

4.1. Cloud Security-SLA

The Cloud Security-SLA is an agreement between the Cloud provider and customer that describes, using generic statements or numerical values, how and how much the CSP will ensure the security of user’s data and services. In this agreement, we should mainly find a specification of the deployed security services on the provider’s side, along with the implemented security techniques and mechanisms, and a definition of their performance levels, usually done with the help of what we called Security Service Level Objective (SSLO) parameters. The Cloud Security-SLA could be hierarchically modeled as illustrated in Fig. 3. In current practices, CSPs do not specify the level of their deployed security services, mainly due to the lack of standard vocabularies and adequate quantitative parameters that express this level, and also to avoid transparency. We propose in this paper, that the Security-SLA should include measurable security parameters, along with the qualitative ones, such as regulations and legal restrictions for data processing and storage. These quantitative parameters should also be standardized in order to facilitate the evaluation and comparison of security of different Cloud infrastructures.

The Security-SLA should cover all components of the offered Cloud service and their associated vulnerabilities and threats. The architecture of the Cloud IaaS service delivery model is depicted in Fig. 4, and Table 1 shows some of the common threats identified for each component of the architecture. For instance, the virtualization layer, which sits between the Operating System and all the other components and enables the partitioning of resources on the hardware level into shared virtual resources between multi-tenant users, is accompanied by many vulnerabilities within the virtual machines and hypervisor. Some of these vulnerabilities are: the possible formation of covert channels between co-located VMs, unrestricted management of the resources by the VMs, and uncontrolled VM migration [25]. Moreover, data-related vulnerabilities such as its co-location, incomplete deletion, and insecure transfer between the different components could also cause the occurrence of dangerous threats and incidents. To assure their customers, CSPs should describe in their Security-SLAs the different security services that they deploy in order to manage these vulnerabilities and ensure protection against the associated threats, since they are solely responsible for the security of the Cloud IaaS service components. The fundamental security attributes of the CIA triad (Confidentiality, Integrity, and Availability) need to be considered when implementing these security services, as we mentioned in our previous work [11]. The confidentiality attribute concerns with protecting the customer’s sensitive information and confidential data from unauthorized disclosure; the integrity attribute consists on protecting the accuracy and validity of data and computations; and the availability attribute describes the continuous accessibility to the Cloud service and data.

The performance of these security services can be described by the means of relevant and measurable SSLO parameters. To derive these parameters, the following set of criteria can be used: accuracy, functionality, correctness, robustness, coverage, continuity, resource overhead, response time, and failure impact, in order to compensate for the difficulty of exploring the indicators that quantify the security level of the Cloud system. In this paper, the GQM method is used to develop the set of SSLO parameters based on a deep analysis of the Cloud IaaS architecture and security services. This method was originally designed to transform the qualitative testing of software

<table>
<thead>
<tr>
<th>Architecture component</th>
<th>Threats</th>
</tr>
</thead>
</table>
| Virtualization         | - Cross-VM attack via Side Channels  
                          - VM hopping or escape  
                          - Insecure VM migration |
| Data storage           | - Data leakage or manipulation  
                          - Data loss  
                          - Data scavenging |
| Network                | - Denial of Service  
                          - Sniffing/Spoofing of virtual networks  
                          - Malware injection |
security to an empirical and measurable testing model. Centering the evaluation process on the security services will permit the comparison of CSP’s security levels and allow for future Security-SLA standardization. With GQM, we define a measurement model on three levels as shown in Fig. 5: on the conceptual level, we describe the goal of the measurement; on the operational level, we characterize the achievement of this goal by generating a set of questions; and finally on the quantitative level, we develop a set of measurable parameters that answer these questions [5]. A set of security services and techniques is presented in Table 2, along with the developed SSLO parameters. Inspired by the three types of measurements [26] that NIST had set to evaluate the performance of an information security system (implementation, effectiveness, and impact), we divide the proposed SSLO parameters into the following categories based on their role:

- **Implementation SSLO parameters.** They intend to reflect the implementation and deployment of appropriate security mechanisms, policies, controls, and procedures on the different layers of the Cloud architecture. They normally take a binary or boolean value: 1 (or yes) if the procedure is deployed, and 0 (or no) if not.

- **Performance SSLO parameters.** They help in monitoring the performance and effectiveness of the provisioned Cloud security services. This SSLO parameter is usually a real number or percentage that expresses a benefit aspect of security. Therefore, security is usually high when the value of such parameter is high.

- **Cost SSLO parameters.** They express the cost of security implementation and the effect of deploying security mechanisms on the performance of the Cloud service. This SSLO parameter is normally a real number or percentage, usually expressing a cost aspect of security, such as the Cloud failure rate. Therefore, security is usually low when the value of such metrics is high.

Normally, when the value of a specific SSLO parameter which was agreed upon in the defined Security-SLA is not met by the CSP, a violation might occur and the customer would have the right to request a compensation as a penalty (e.g., financial). However, in this paper, violations of the Security-SLA will be measured according to the global satisfaction of SSLO parameters, as we will see next.

### 4.2. Security Level Evaluation

The development of SSLO parameters was the first step towards quantifying the Cloud security level. In this section, we first propose a method to evaluate the security level of CSPs. Then, taking into consideration their security reputations, which we compute relatively to their customers’ satisfaction, we evaluate the security levels of the federations that could form between them. To evaluate the security levels of CSPs and generate a single numerical value that reflects their security status, we apply a coarse grained aggregation method that is based on comparing their provided SSLO values to a predefined baseline level, defined as follows.

**Definition 1.** Cloud Security-SLA Baseline (CSB). It is a version of the Security-SLA that expresses the minimum required security level that a CSP should achieve when providing Cloud services.

The CSB could be set according to the types of services (data storage, web service, scientific computing, financial simulations, etc) based on their security requirements (e.g., data sensitivity), and is determined by third party security experts. The baseline security level is expressed using the same SSLO parameters that we developed, and can be constructed by carefully considering previous Cloud customers’ experiences with deploying services to the Cloud. For instance, if an implementation SSLO parameter was found to be not fully essential for the protection of the Cloud service, its value will remain 0. Information related to the CSB will be stored on the broker side and updated occasionally when new Cloud threats are discovered or new SSLO parameters need to be added. Therefore, the security level of CSPs is continuously evaluated against the baseline level to demonstrate compliance and keep the information stored on the brokers’ side up to date. When the security baseline is defined and set, the security level of CSPs and their formed federations will be evaluated by measuring their deviation from the baseline level. We assume that the federations will form between the CSPs providing the same type of services, and therefore, the members will be evaluated against the same CSB (in the case where multiple CSBs exist). The principle motivation behind the definition of a security baseline is to build a reference level that will facilitate the evaluation and comparison of CSPs’ security levels and help standardizing the security evaluation process in the Cloud. Another reason is that CSPs will usually share their resources to service several customer requests, therefore, the security requirements of those requests can not from the basis of security-based federation formation.
Table 2: A set of SSL0 parameters related to each security service in the Cloud Security-SLA. Parameter type I refers to implementation, P to performance, and C to cost.

<table>
<thead>
<tr>
<th>Security service</th>
<th>Security techniques and mechanisms</th>
<th>SSL0 parameters</th>
<th>Parameter type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity and Access Management</td>
<td>- False acceptance and rejection rates of authentication mechanisms</td>
<td>- Configuration of access control lists (ACLs) on virtual interface ports</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Enforcement of policies on password strength and expiration</td>
<td>- Intrusions and injection attacks detection success rate</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>- Blocking of invalid login attempts</td>
<td>- Intrusion detection false positives</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Enabling of client certificate for SSL/TLS</td>
<td>- Mean-time to discover and mitigate attack</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Average response time of transactions</td>
<td>- Service latency or response time during attack</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Implementation of risk-based entitlement decisions</td>
<td>- Network packets drop rate during attack</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Ability to use temporary access credentials</td>
<td>- CPU load or processing usage of IDS</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Frequency of review of system users and administrators’ entitlements</td>
<td>- Network load or bandwidth overhead of IDS</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Frequency of review of access control logs and accounts’ activity</td>
<td>- Frequency of network penetration tests</td>
<td>C</td>
</tr>
<tr>
<td>Network security</td>
<td>- Configuration of security groups</td>
<td>- Capability of open encryption methodologies</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>- Configuration of Access Control Lists (ACLs) on virtual interface ports</td>
<td>- Capability of creation of a unique encryption key per tenant</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>- Intrusions and injection attacks detection success rate</td>
<td>- Internal storage of encryption keys</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>- Intrusion detection false positives</td>
<td>- Encryption key length</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>- Mean-time to discover and mitigate attack</td>
<td>- Enabling of HTTP Strict Transport Security (HSTS)</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>- Service latency or response time during attack</td>
<td>- Database deployment with SSL protected transactions</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>- Network packets drop rate during attack</td>
<td>- Support of secure data deletion</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>- CPU load or processing usage of IDS</td>
<td>- Implementation of data loss/leakage prevention techniques</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>- Network load or bandwidth overhead of IDS</td>
<td>- Percentage of key storage overhead</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>- Frequency of network penetration tests</td>
<td>- Storage node online latency in responding to read-write requests</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- CPU load or processing usage of IDS</td>
<td>- Data backup frequency</td>
<td>P</td>
</tr>
<tr>
<td>Data and storage security</td>
<td>- Capability of open encryption methodologies</td>
<td>- Backup restoration time</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Capability of creation of a unique encryption key per tenant</td>
<td>- Number of redundant backup sites</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Internal storage of encryption keys</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Encryption key length</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Capability of encryption of virtual storage</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Validation of VMs</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Implementation of encrypted live migration</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Database deployment with SSL protected transactions</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Support of secure data deletion</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Implementation of data loss/leakage prevention techniques</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Percentage of key storage overhead</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Storage node online latency in responding to read-write requests</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Virtualization security</td>
<td>- VMs’ interference prevention</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Hypervisor-level role-based access control</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- VM encryption</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- SSH secure communications</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Malware detection</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Secure VM migration</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Events monitoring and auditing by hypervisor</td>
<td>- Frequency of assessment of virtualization vulnerabilities</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>- Capability of encryption of virtual storage</td>
<td>- VM backup, restoration and clean-up capabilities</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>- Validation of VMs</td>
<td>- Slowdown in migration time due to secure migration</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Implementation of encrypted live migration</td>
<td>- Performance overhead due to running filters on VM images</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>- Data destruction procedures after migration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical security</td>
<td>- Events monitoring and auditing by hypervisor</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Frequency of assessment of virtualization vulnerabilities</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- VM backup, restoration and clean-up capabilities</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Slowdown in migration time due to secure migration</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Performance overhead due to running filters on VM images</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Frequency of assessment of virtualization vulnerabilities</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- VM backup, restoration and clean-up capabilities</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Slowdown in migration time due to secure migration</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Performance overhead due to running filters on VM images</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Frequency of assessment of virtualization vulnerabilities</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- VM backup, restoration and clean-up capabilities</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data and Computational</td>
<td>- Deployment of applications with computing replication</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td>integrity</td>
<td>- Encryption of VM images during transport across hypervisor instances</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Computation time by PDP techniques</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Bandwidth overhead of PDP tokens’ transmission</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Processing overhead of accountable computation</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- False positives of computational faults</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Deployment of applications with computing replication</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td>Data and service availability</td>
<td>- Multi-failure disaster recovery capability</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Capability of infrastructure service fail-over to other providers</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Configuration of SIEM incident reporting, analysis and alerting</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Frequency and coverage of risk analysis</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Frequency of reviewing and testing of business continuity plans</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Data redundancy level</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Mean-time of incident discovery and recovery</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Average time between incident detection and reporting</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Percentage of recurring incidents</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Cloud failure rate and failure repair time</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Time to recover failed tasks</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- System downtime due to incidents</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- FRC and DDoS attack detection time</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Time to identify attack source</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Processing overhead of detection mechanisms</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Application response time under attack</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>- Mean-time to mitigate or patch vulnerability</td>
<td></td>
<td>C</td>
</tr>
</tbody>
</table>

Thus the need for a flexible reference evaluation model. The notations used in the evaluation of the security level are presented in Table 3. The deviation of a Cloud service provider $CSP_n \in CSP$ from the baseline level, denoted by $\alpha_{n,t}$, is computed as follows:
where \( J \), \( K \), and \( L \) are respectively the numbers of implementation, performance, and cost SSLO parameters defined in the security-SLA, and each term of Eq. (2) reflects the average weighted relative deviation of the respective type of SSLO parameters from the baseline. To homogenize the evaluation, the values of SSLO parameters offered by CSPs are normalized with respect to the values given in the baseline, since these parameters have different units of measurement. According to the equation, the deviation from the baseline level is positive when the weighted average deviations of each parameter type are positive (or their sum is positive), i.e., the number of satisfied implementation parameters is higher than the required one, the majority or all performance parameters have higher values than what it is required by the baseline, and the majority or all cost parameters have lower values than what it is required by the baseline. The security level of \( CSP_n \), denoted by \( SL_n \), will be then expressed as follows:

\[
SL_n = 1 + \alpha_n
\]  

(2)

A positive value of \( \alpha_n \) reflects a security level that is higher than the baseline level, whereas a negative value of it expresses a security level that is lower than the baseline level. We consider a CSP secure if it provides a security level that is higher than or equal to the baseline level (e.g., \( \geq 1 \)), and insecure otherwise. For instance, considering a percentage like evaluation result, we would say that \( CSP_n \in CSP \) has a security level of 20% over or under the defined baseline if \( SL_n = 1.2 \) or 0.8 respectively. Therefore, \( CSP_n \) is considered to be secure in the first case and insecure in the second.

The weights of SSLO parameters in Eq. (1) aim at determining their relative significance to the evaluation process, and are computed with the help of security experts based on several factors such as the Cloud service type. For instance, if the deployed Cloud service is a web server, the SSLO parameter “System downtime due to incidents” is more important than the SSLO parameter “False positives of computational faults” since the availability of the service in this case is more critical to the business than the integrity aspect.

Each CSP interacts with its customers by servicing their requested sets of workloads, which could be represented by a number of VMs of different sizes in terms of the offered resources. Each workload entails a set of security requirements that are mapped to specific SSLO values. This mapping can be performed by transforming the qualitative description of customer’s requirements into the quantitative parameters of table 2. For instance, to ensure service availability, the requirement “Disruption of Cloud services” is proposed, along with five possible descriptive options: absolutely intolerant, intolerant, moderately tolerant, and absolutely tolerant. These qualitative values can then be mapped to the quantitative SSLO parameter “System downtime due to incidents” by defining the correspondent intervals of service downtime in hours according to the service’s characteristics and nature. What matters in our case, is that a requested Security-SLA will be constructed for each of the customer’s workloads, including all SSLO values which it defined. Then, a required security level is computed for each workload relatively to the CSB using Eq. (1) and (2).

### 4.3. Federation Security Evaluation

To evaluate the security level of a Cloud federation, we first start by evaluating the reputation of CSPs with respect to security, which is a parameter that reflects the degree to which CSPs are providing their customers with the promised security level. Let \( CSC = \{ CSC_h, h \in N^*, h \leq H \} \) denote the set of \( H \) Cloud Service Customers that are interacting with a specific CSP, and each customer requests the servicing of a set of workloads \( W_h \). The security of a workload is considered to be satisfied by the CSP if, and only if, the security level that was provided by the CSP is higher than or equal to the workload’s required security level, which was promised by the CSP. Thus, if we assume that customer \( CSC_h \) will only interact with the CSP if the latter had promised to satisfy the security levels required by its workloads, the security satisfaction \( Sat_h \) of the customer can be computed by simply dividing the number of workloads which required security was satisfied when serviced by the CSP by the total number of workloads during the interaction, as follows:

\[
Sat_h = \frac{|W_{h}^{Sat}|}{|W_h|} 
\]  

(3)

where \( W_{h}^{Sat} \subseteq W_h \) and \( W_{h}^{Sat} \) is the set of workloads which required security was satisfied by the CSP. Finally, in order to compute the reputation value \( rep_n \) of a Cloud service provider \( CSP_n \) that is attributed to its security level, and which will indicate how much the CSP is being honest about the promised security levels, the security satisfaction of all its customers is considered. The following equation evaluates \( rep_n \) by averaging the security satisfaction of all the \( H \) customers of \( CSP_n \):

\[
rep_n = \frac{1}{H} \sum_{h=1}^{H} Sat_h
\]  

(4)

Before evaluating the security level of a federation, we start by formally defining the federation formation concept in the context of coalition formation games as follows.

### Definition 2. Coalition partition. A coalition structure or partition is a set of \( M \) coalitions \( \Pi = \{ F_m, m \in N^*, m \leq \)
where each $F_m \subseteq CSP$ is a disjoint coalition such as $\bigcup_{m=1}^{M} F_m = CSP$ and $F_m \cap F_y = \emptyset \forall y \neq m$.

The coalitions in our case are called federations. When forming a federation, we assume that the CSPs are already fulfilling the constraints on workload servicing, that is, they are able to provide sufficient resource capacities, and we look beyond this criterion to verify the security status of the members. We define the security level function $Sec$ for the formed federations as a real-valued function such that $Sec : F_m \rightarrow \mathbb{R}^+$ as follows:

$$Sec(F_m) = \sum_{CSPr_n \in F_m} R_n rep_n SL_n \sum_{CSPr_n \in F_m} R_n$$

(5)

This function aims at computing the security level of a formed federation based on the security levels and reputations of its members. Multiplying the reputation value of a CSP by its declared security level instead of its value of a CSP by its declared security level instead of its

Federation, since all its VMs are provided with the level of Security-SLA, regardless of their types or sizes. Consequently, all CSPs in the federation $F_m$ will provide their federated services with the same security level $Sec(F_m)$, since they are sharing the federation’s resources.

5. The Security-based Cloud Federation Formation Game

When deciding to form a Cloud federation and sharing their resources, CSPs should consider the security factor, which will lead them to forming secure federations and reducing Security-SLA violations. Coalition formation is a major subject in multi-agent systems, and hedonic games are a popular category of the coalitional cooperative games, and in which profit allocation among the coalition members is not the main problem. In a hedonic game, the players are usually self-interested, and the stability property is guaranteed, that is, when the final partition of coalitions is formed, none of the players will have an incentive to leave its current coalition to join another. In the case of Cloud federation formation, the number of possible coalition structures is too large to permit an exhaustive search for the optimal solution and finding it is a NP-complete problem [27]. Thus, a hedonic game was adopted to model the problem. In this section, the security-based Cloud federation formation process is modeled as a hedonic coalitional cooperative game that aims at producing secure federations, and where each CSP acts as a selfish player when deciding to prefer a federation over another according to the security levels of its members. The appropriate preference function is defined and the properties of the game are analyzed.

5.1. Game Model

A coalitional game is a game-theoretical approach that models the interactions among players when they aim at forming groups, and generates a partition of coalitions over the set of players. In the game that we propose, the players are the Cloud service providers and the coalitions to be formed are the Cloud federations. The objective of the players is to join secure federations where the number of insecure CSPs is minimal. When the federation size increases, the probability of finding insecure members in the federation is higher. We introduce the following definitions.

Definition 3. Grand federation. The grand federation $G$ is formed when all CSPs decide to join one single federation, that is, $G = CSP = \{CSPr_1, \ldots, CSPr_N\}$.

Definition 4. Non-cohesive game. A coalitional game is considered to be non-cohesive [28] when producing a set of disjoint coalitions is preferred over forming a single coalition that groups all the players together.

In our game, forming the grand federation will entail higher drops in the security levels of its members and increase the rates of Security-SLA violations, since secure and insecure CSPs will exist in the same federation. Therefore, we consider our game to be non-cohesive which goal is to generate a set of disjoint federations that separate secure CSPs from insecure ones. In contrary to Transferable Utility (TU) coalitional games where the utility of the coalition can be transferred and distributed among its members [6], our proposed game is considered of Non-Transferable Utility (NTU) since security cannot be distributed among the federation members.

Definition 5. Hedonic game. A hedonic game is a type of NTU coalitional games where the utility of a player in a coalition depends only on its members, and the formation of coalitions is based on the preferences that the players have over the set of possible coalitions [6].

We consider our game to be hedonic since it verifies these two conditions. First, the security level of a CSP in a particular federation depends only on the CSPs that are members in that federation as shown in Eq. (5), and second, we look at security as an enjoyable property which CSPs will consider as the basis for building their preferences over the set of federations.

At any given time, the set of CSPs is divided into a partition $\Pi = \{F_m, m \in \mathbb{N}, m \leq M\}$ with $M$ federations, where each $F_m \subseteq CSP$ is a disjoint federation such as $\bigcup_{m=1}^{M} F_m = CSP$ and $F_m \cap F_y = \emptyset \forall y \neq m$.

Given a federation partition $\Pi$, for any $CSPr_n \in CSP$, we denote by $F_{II}(n)$ the federation $F_m \in II$ such that $CSPr_n \in F_m$. Each coalition possesses a coalition value, which in our case is the security level of the formed federation that we evaluate using Eq. (5) relatively to the defined CSB by taking into account the security levels of the federation members and the amount of resources each member shares with the federation. The Cloud federation formation game proposed in this paper is defined as follows.

Definition 6. The security-based Cloud federation formation
game. It is the pair \((CSP, \succeq)\), where CSP is the set of Cloud Service Providers in the game, and \(\succeq\) is a reflexive, complete, and transitive preference relation on the set of all federations that CSP\(_n\) can form based on the security level.

Based on this definition, for all CSP\(_n \in CSP\) and for all \(E, E' \in \Pi\), we define \(\succeq_n\) as

\[
E \succeq_n E' \iff U_n(E) \geq U_n(E') \tag{6}
\]

where \(U_n(E)\) and \(U_n(E')\) are the utilities of CSP\(_n\) in federations \(E\) and \(E'\) respectively. The utility function can be defined as follows:

\[
U_n(E) = \begin{cases} 
Sec(E) & \text{if } E \notin h_n \\
0 & \text{otherwise} 
\end{cases} \tag{7}
\]

where \(h_n\) is a history set where CSP\(_n\) stores the identities of the federations which was a part of in the past. Hence, the preference relation assigns a utility value of 0 for any federation that already exists in \(h_n\), in order to avoid visiting the same federation more than one time during the federation formation process (a similar concept has also been adopted in [22], [29], [30]). Based on the defined preference relation, each CSP will compare the set of possible federations and indicate its intention to be a part of one of them, which means that it will prefer the federation that will grant the higher security level and avoid joining federations with insecure members. \(E \succeq_n E'\) indicates that CSP\(_n\) prefers to be a member of federation \(E\) than to be a member of federation \(E'\), or at least prefers them both equally. The strict counterpart of the relationship denoted by \(\succ_n\) implies that CSP\(_n\) strictly prefers to be a member of \(E\) over \(E'\).

5.2. The Federation Formation Algorithm

In this section, we propose a hedonic federation formation algorithm that achieves the goal of our security-based federation formation game, and that can be implemented in a distributed fashion. The algorithm is illustrated in Algorithm 1, and takes as input the initial federation partition \(\Pi_c\), the set \(R\) of resources that CSPs will share with the federations, and the security levels of CSPs computed using Eqs. (1) and (2), and outputs the final federation partition \(\Pi_f\).

The following hedonic shift rule [30], which is selfishly executed by each CSP when moving between federations, is used as the basis of the proposed federation formation algorithm: given a federation partition \(\Pi_c\) on the set of service providers CSP and a preference relation \(\succ_n\), any CSP\(_n \in CSP\) decides to leave its current federation \(F\(_\Pi(n)\)\) to join another one \(F_m \in \Pi_c \cup \emptyset\) if, and only if, \(F_m \cup \{CSP\_n\} \succ_n F\(_\Pi(n)\)\), that is, if the utility of CSP\(_n\) in the new federation is higher than its utility in the current one. In other words, if the security level of the new federation is higher than the security level of the current one.

The actions presented in the algorithm are executed asynchronously and independently by each CSP during the federation formation process. Starting from a federation partition \(\Pi_c\), each service provider CSP\(_n\) evaluates the utility that it will acquire by joining a new federation \(F_m\) and compares it to the one it has from its current federation \(F\(_\Pi(n)\)\), then applies the described shifting rule if the utility of CSP\(_n\) in the new federation \(F_m\) exceeds its utility in \(F\(_\Pi(n)\)\), and automatically updates its history set \(h_n\) by adding the federation \(F\(_\Pi(n)\)\) that it had left (lines 6 to 12). Otherwise, CSP\(_n\) remains in its current federation and the partition \(\Pi_c\) remains unchanged. The federation formation procedure is repeated until all CSPs converge to a final partition \(\Pi_f\) that satisfies Nash-stability.

The execution of the algorithm is repeated periodically to reflect the updates of CSPs’ security levels, reputation values, and the set of their available resources. However, it is deemed necessary to provide the appropriate mechanisms for: state retrieval (e.g., [31]), which permits to each CSP to obtain the current federation partition, and atomic state update (e.g., [32]), which guarantees that the current federation partition will remain unchanged while the CSP is executing the actions in the algorithm.

The main computational complexity of the proposed algorithm lies in the hedonic shift rule that the CSP will apply to switch from a federation to another, and depends on the number of federations in the partition \(\Pi_c\). Let \(|\Pi_c|\) be the number of federations in \(\Pi_c\), the computational complexity can then be expressed by \(O(|\Pi_c|)\). The worst computational complexity will occur in the case where \(\Pi_c\) contains \(N\) federations, that is, every CSP is acting alone in a separate coalition. Since the federation formation procedure is repeated until reaching the final partition, the computational complexity of the game will be defined by the Bell number in the worst case scenario (discussed in the following subsection).

5.3. Game Analysis

In this section, the properties of the proposed security-based federation formation game are analyzed. More specifically, we demonstrate the property of convergence of the federation formation algorithm to a final solution and the property of stability of the produced final solution.

Definition 7. Nash-Stability. A partition of federations \(\Pi\) is considered Nash-stable if no CSP has an incentive to move from

---

Algorithm 1 The security-based Cloud federation formation algorithm.

**Input:**
- The current federation partition \(\Pi_c = \{F_1, \ldots, F_M\}\)
- The set \(R = \{R_1, \ldots, R_N\}\) of available resources
- The set \(SL = \{SL_1, \ldots, SL_N\}\) of CSPs’ security levels

**Output:**
- The final federation partition \(\Pi_f\)

1: procedure FEDERATIONFORMATION(\(\Pi_c, R, SL\))
2: \(h_n \leftarrow \emptyset\) empty set
3: for all CSP\(_n \in CSP\) do
4: for all federations \(F_m \in \Pi_c \cup \emptyset\) do
5: evaluate \(Sec(F_m \cup \{CSP\_n\})\)
6: if \(U_n(F_m \cup \{CSP\_n\}) > U_n(F\(_\Pi(n)\))\) \(\iff F_m \cup \{CSP\_n\} \succ_n F\(_\Pi(n)\)\) then
7: - leave the current federation \(F\(_\Pi(n)\)\)
8: - join the new federation \(F_m \cup \{CSP\_n\}\)
9: - update the federation partition:
10: \(\Pi_{c+1} = (\Pi_c \setminus \{F\(_\Pi(n)\), F_m\}) \cup \{F_m \cup \{CSP\_n\}\}\)
11: \(\Pi_{c+1} = (\Pi_c \setminus \{F\(_\Pi(n)\), F_m\}) \cup \{F_m \cup \{CSP\_n\}\}\)
12: - Update history set: \(h_n = h_n \cup \{F\(_\Pi(n)\)\}\)
13: else
14: CSP\(_n\) remains in \(F\(_\Pi(n)\)\):
15: \(\Pi_{c+1} = \Pi_c\)
16: end if
17: end for
18: end for
19: return \(\Pi_f = \Pi_{c+1}\)
20: end procedure
its current federation \(F_{\Pi}(n)\) to join a different one, nor to act alone. That is, for \(\Pi = \{F_m, m \in \mathbb{N}^n, m \leq M\}, \forall CSP_n \in CSP, F_{\Pi}(n) \supseteq_n F_m \cup \{CSP_n\} \forall F_m \in \Pi \cup \emptyset.

**Definition 8. Individual Stability.** A partition of federations \(\Pi\) is considered individually stable if no CSP can benefit \(\Pi\) without negatively affecting the members of the latter. That is, \(\Pi\) is individually stable if there is no CSP that prefers to leave its current federation and join either one of the other two federations, and join a new one, every member of the latter federation will prefer to stay non-cooperative over joining \(CSP_1\) or \(CSP_2\) in a federation. Therefore, the individual stability in the final partition is achieved.

**Proposition 1. Convergence.** Starting from any initial partition \(\Pi_0\), Algorithm 1 converges to a final partition \(\Pi_f\) consisting of a number of disjoint federations.

**Proof.** Every application of the hedonic shift rule will transform the current partition \(\Pi_c\) into a new partition \(\Pi_{c+1}\) that has not been visited in the past (according to Eq. (7)), until reaching the final partition \(\Pi_f\). Since the number of transformations is finite, and at most, is equal to the number of partitions defined by the \(N\)th Bell number \(B_n\) for \(N\) CSPs as follows:

\[
B_n = \sum_{i=0}^{N-1} \binom{N-1}{i} B_i \quad \text{for } N \geq 1 \text{ and } B_0 = 1
\]

the sequence of transformations will always be limited and will converge to a final partition \(\Pi_f\).

**Proposition 2. Nash-Stability.** Any final partition \(\Pi_f\) produced by Algorithm 1 is a Nash-stable federation partition.

**Proof.** This proposition can be proved by contradiction. Let us assume that the final partition of federations \(\Pi_f\) is not Nash-stable. Then, there exists a service provider \(CSP_n\) that prefers to leave its current federation \(\Pi_f(n)\) and join another one \(F_m\), that is, \(F_m \cup \{CSP_n\} \succ_n \Pi_f(n)\). Therefore, \(CSP_n\) will perform the shifting rule and the final partition will change to a new one \(\Pi_f^1\) such as \(\Pi_f^1 \neq \Pi_f\) which contradicts with Proposition 1 that states that the partition \(\Pi_f\) is the final outcome of the federation formation algorithm. Hence, we conclude that the algorithm always converges to a Nash-stable federation partition.

**Proposition 3. Individual Stability.** Any final partition \(\Pi_f\) that is produced by Algorithm 1 is individually stable.

**Proof.** Since we have already proven that Algorithm 1 converges to a Nash-stable partition of federations, this implies that it also converges to an individually stable partition.

To illustrate the individual stability of the partitions generated by our proposed game, the following numerical example is considered. We study the stability of a security-based federation formation game between 7 players such that \(SL = \{0.72, 1.1, 1.3, 0.55, 1.4, 1.2, 0.5\}\) and \(R = \{100, 200, 150, 320, 240, 170, 450\}\). The initial partition of federations is shown in the first row of Table 4. Every row shows the partition \(\Pi_{f+1}\) that is generated at the end of each execution of Algorithm 1 by each of the CSPs, and the last row shows the final partition \(\Pi_f\). As we notice in \(\Pi_{f+1}\), CSP1 and CSP4 both have incentive to leave their current federation and join one of the other two federations, since \(\{CSP_1\} \cup \{CSP_4\} \not\succ_1 \{CSP_2, CSP_3\}\). If one of them decides to move from its current federation and join a new one, every member of the latter federation will be worsen off. For instance, the federation \(\{CSP_2, CSP_3, CSP_4, CSP_5\}\) is preferred by each of its members over the federations \(\{CSP_3, CSP_5, CSP_6, CSP_7, CSP_8\}\) and \(\{CSP_3, CSP_5, CSP_6, CSP_7\}\) since it is more secure. Similarly, \(CSP_2\) will prefer to stay non-cooperative over joining \(CSP_1\) or \(CSP_4\) in a federation. Therefore, the individual stability in the final partition is achieved.

### 6. Experimental Results and Analysis

In this section, we study the performance of the security-based Cloud federation formation game. First, we explain the experimental setup used in our simulation, then we analyze the results. The goal is to show that when CSPs consider the security factor when forming federations, they tend to form federations that are more secure and refrain from joining federations that will cause high security loss. Consequently, security-based federation formation will reduce the rate of Security-SLA violations and their severity.

#### 6.1. Experimental Setup

The security-based federation formation game is implemented on a 64-bit Windows 7 machine equipped with an Intel Core i7-361QM CPU @ 2.10 GHz Processor and 12 GB RAM. Since we haven’t found in the literature any other approach that discussed Cloud federation formation based on security, and we found that other
existing approaches are not directly comparable to ours, we decided to compare our model with the following two models: the grand federation model which consists on forming the grand federation between all CSPs without any consideration to any factor, and the random federation formation model which consists on randomly grouping CSPs together and forming federations of relatively bigger size than when considering a specific factor during formation (e.g., security). Throughout the simulation, we vary the percentage of insecure CSPs from 10% to 70% and compare the performance of the three models, starting from a randomly generated initial partition of federations. We implement the security-based federation formation algorithm using MATLAB, where CSPs are simulated as objects, each having an offered security level, a required security level defined by its customers, a reputation value, and a number of VMs to be shared with a federation.

Cloud Security-SLA is still a concept that needs to be continually developed to reach standardization. Security-SLAs do not exist today in the quantitative form that we propose in this paper, but they might exist, and not widely, in a more primary version. Either way, the transparency of CSPs about what they state and include in their Security-SLAs is relatively shy, which makes it challenging to collect information related to security deployment and performance. Therefore, we decided to generate our own data for the sake of these experiments. The values of SSLO parameters for each CSP and for the baseline level, along with other experimentation parameters, are randomly generated as shown in Table 5. The experiment is done with a total of 100 CSPs and the evaluation of security level is performed based on a total of 120 parameters divided between the three categories: implementation, performance, and cost. To reduce complexity, we assume that all parameters have the same importance to the evaluation process, hence parameters’ weights are not considered in our experiments.

The two main aspects that we consider in our performance study are the effect of the security-based federation formation on the initial security levels of CSPs, and the Security-SLA violations caused by the federation formation. We aim at showing that our model reduces the formation and compare the performance of the three models, starting from a randomly generated initial partition of federations. We implement the security-based federation formation algorithm using MATLAB, where CSPs are simulated as objects, each having an offered security level, a required security level defined by its customers, a reputation value, and a number of VMs to be shared with a federation.

Cloud Security-SLA is still a concept that needs to be continually developed to reach standardization. Security-SLAs do not exist today in the quantitative form that we propose in this paper, but they might exist, and not widely, in a more primary version. Either way, the transparency of CSPs about what they state and include in their Security-SLAs is relatively shy, which makes it challenging to collect information related to security deployment and performance. Therefore, we decided to generate our own data for the sake of these experiments. The values of SSLO parameters for each CSP and for the baseline level, along with other experimentation parameters, are randomly generated as shown in Table 5. The experiment is done with a total of 100 CSPs and the evaluation of security level is performed based on a total of 120 parameters divided between the three categories: implementation, performance, and cost. To reduce complexity, we assume that all parameters have the same importance to the evaluation process, hence parameters’ weights are not considered in our experiments.

The two main aspects that we consider in our performance study are the effect of the security-based federation formation on the initial security levels of CSPs, and the Security-SLA violations caused by the federation formation. We aim at showing that our model reduces the formation and compare the performance of the three models, starting from a randomly generated initial partition of federations. We implement the security-based federation formation algorithm using MATLAB, where CSPs are simulated as objects, each having an offered security level, a required security level defined by its customers, a reputation value, and a number of VMs to be shared with a federation.

Cloud Security-SLA is still a concept that needs to be continually developed to reach standardization. Security-SLAs do not exist today in the quantitative form that we propose in this paper, but they might exist, and not widely, in a more primary version. Either way, the transparency of CSPs about what they state and include in their Security-SLAs is relatively shy, which makes it challenging to collect information related to security deployment and performance. Therefore, we decided to generate our own data for the sake of these experiments. The values of SSLO parameters for each CSP and for the baseline level, along with other experimentation parameters, are randomly generated as shown in Table 5. The experiment is done with a total of 100 CSPs and the evaluation of security level is performed based on a total of 120 parameters divided between the three categories: implementation, performance, and cost. To reduce complexity, we assume that all parameters have the same importance to the evaluation process, hence parameters’ weights are not considered in our experiments.

The two main aspects that we consider in our performance study are the effect of the security-based federation formation on the initial security levels of CSPs, and the Security-SLA violations caused by the federation formation. We aim at showing that our model reduces the formation and compare the performance of the three models, starting from a randomly generated initial partition of federations. We implement the security-based federation formation algorithm using MATLAB, where CSPs are simulated as objects, each having an offered security level, a required security level defined by its customers, a reputation value, and a number of VMs to be shared with a federation.

Cloud Security-SLA is still a concept that needs to be continually developed to reach standardization. Security-SLAs do not exist today in the quantitative form that we propose in this paper, but they might exist, and not widely, in a more primary version. Either way, the transparency of CSPs about what they state and include in their Security-SLAs is relatively shy, which makes it challenging to collect information related to security deployment and performance. Therefore, we decided to generate our own data for the sake of these experiments. The values of SSLO parameters for each CSP and for the baseline level, along with other experimentation parameters, are randomly generated as shown in Table 5. The experiment is done with a total of 100 CSPs and the evaluation of security level is performed based on a total of 120 parameters divided between the three categories: implementation, performance, and cost. To reduce complexity, we assume that all parameters have the same importance to the evaluation process, hence parameters’ weights are not considered in our experiments.

The two main aspects that we consider in our performance study are the effect of the security-based federation formation on the initial security levels of CSPs, and the Security-SLA violations caused by the federation formation. We aim at showing that our model reduces the formation and compare the performance of the three models, starting from a randomly generated initial partition of federations. We implement the security-based federation formation algorithm using MATLAB, where CSPs are simulated as objects, each having an offered security level, a required security level defined by its customers, a reputation value, and a number of VMs to be shared with a federation.

Cloud Security-SLA is still a concept that needs to be continually developed to reach standardization. Security-SLAs do not exist today in the quantitative form that we propose in this paper, but they might exist, and not widely, in a more primary version. Either way, the transparency of CSPs about what they state and include in their Security-SLAs is relatively shy, which makes it challenging to collect information related to security deployment and performance. Therefore, we decided to generate our own data for the sake of these experiments. The values of SSLO parameters for each CSP and for the baseline level, along with other experimentation parameters, are randomly generated as shown in Table 5. The experiment is done with a total of 100 CSPs and the evaluation of security level is performed based on a total of 120 parameters divided between the three categories: implementation, performance, and cost. To reduce complexity, we assume that all parameters have the same importance to the evaluation process, hence parameters’ weights are not considered in our experiments.
members increases. Second, the main criteria that should be considered in this case to establish a fair comparison between the security-based and random formation models is the stability of the maximum security level. For instance, the maximum security level obtained in the final partition when applying random formation rapidly decreased when the percentage of insecure members in the initial partition increased, whereas the security-based formation model conserved approximately the same maximum security level (up to 1.42) indifferently from the increase in the percentage of insecure members. This shows that our model will keep generating secure federations by separating secure from insecure CSPs.

The variation in CSPs’ initial security levels between being in a non-cooperative state and being in a federation is directly related to the occurred Security-SLA violations. Fig. 8 illustrates the difference between the three models with regard to respecting the agreed upon Security-SLA. In Fig. 8a, we show the variation in the rates of Security-SLA violations while increasing the percentage of insecure CSPs in the initial partition. The results demonstrate the efficiency of our model in producing federations that guarantee minimum violation rates (between 11 and 15%) comparing to the other two models (between 22 and 40%).

The reason is that violations are correlated with the drop in CSPs’ initial security levels, and usually occur when the security level of the federation where the CSP exists in the final partition is lower than the security level that was promised to its customers when accepting to service their workloads. For example, when the percentage of insecure CSPs reached 60%, the rate of violations was minimum (11%), which corresponds to the minimum loss in security (21%) as shown in Fig. 7a.

Fig. 8b measures the severity of occurred Security-SLA violations when applying the three models, and while varying the percentage of insecure CSPs in the initial partition. We define the violation severity by:

$$\text{Violation Severity} = \frac{\text{provided security level}}{\text{promised security level}}$$  \hspace{1cm} (9)

where the provided security level is the same as the federation security level where the CSP exists in the final partition, and the promised security level is the one that the customer requested for its workload. Even though our model does not completely avoid security violations, it achieves federation formation while keeping a minimum and stable level of violation severity (maximum of 7%), whereas with the other two models, the severity level
continued to increase with the increase in the percentage of insecure CSPs. This is mainly due to the fact that random federation and grand federation formation do not consider the security factor when grouping CSPs into federations, thus, secure CSPs end up joining federations with high number of insecure CSPs which affect the overall security level of the formed federation, and consequently, the provided security level of its members. In real life scenarios, the rate of violations is not as critical as their severity, since the severity of the violation is what controls the applied penalty cost. For instance, considering the cost SSLO parameter “Cloud failure rate”, if the promised value was determined to be not higher than 5%, a provided value of 15% will result in a violation that is more severe than a violation resulted from a provided value of 7%, hence, the cost of the applied penalty will be higher in the first case.

In Fig. 9, the average size of federations in the final partition $\Pi_f$ is illustrated. As we can see, our model tends to group CSPs into federations of small size (up to 2 or 3 CSPs per federation), in contrast to forming the grand federation which will group all CSPs together in one single federation. The intention from measuring the size of the final formed federations is to show that our model distributes the CSPs onto federations according to their security levels, where CSPs of relatively close security levels end up together in the final federation. For instance, if the average federation size in the final partition was higher than the one showed in the figure, the probability to encounter insecure CSPs in a federation of secure ones would increase. Therefore, by considering the security factor, the proposed federation formation algorithm works in a way that conserves a stable federation size and refrains from grouping insecure CSPs with secure ones.

Finally, we aim at studying the execution time of the game. In Fig. 10, we present the average results of fifty runs along with the standard deviations with respect to the number of CSPs involved in the game. We notice that the computational time is relatively small comparing to an exhaustive search approach. For instance, when the number of CSPs was equal to 8, the algorithm was exploring 57 federations on average instead of all the 255 possible federations before converging to the final partition. This demonstrates the scalability of our model when applied on a high number of players.

We also find it necessary to discuss other security and performance factors that would be influenced by the application of our proposed security-based federation formation model, and which we did not numerically consider in our experiments. For instance, by considering the security levels of CSPs when forming Cloud federations, the shared environment which CSPs will create in their federation will somewhat guarantee the same availability level that CSPs were providing in their respective Security-SLAs, along with the other security aspects that CSPs agreed on protecting. In addition, attack propagation will be limited since secure CSPs are now federating their workloads to each other, and sharing their performance and expertise in attack detection and mitigation, away from insecure environments where the probability of a threat occurrence is higher. Consequently, security risk will also be reduced, since the risk is directly proportional to the probability of threat occurrence [33]. Therefore, we can conclude that security-based Cloud federation formation will allow the federation members to safely and peacefully enjoy each other’s company and share their resources in a more secure and efficient fashion.

7. Conclusion

In this paper, the Cloud federation formation problem was addressed from a new perspective. The originality of our work lies within three aspects: the quantitative Cloud Security-SLA parameters that we developed, the proposed method that evaluates the security level of CSPs and federations, and the security-based Cloud federation formation game that we modeled and studied. It is also important to note, that considering Cloud customers’ security satisfaction and CSPs’ reputation during the evaluation process introduced an added value to the accuracy of the model by compensating for the miss-leading information about CSPs’ offered security levels. We found it fair enough to consider CSPs’ security levels along with their provided shares of the federation resources when evaluating the security level of a federation. The proposed federation formation algorithm enabled the CSPs that offer relatively close security levels to be grouped together in the same federation, while reducing their loss in security and minimizing the rates and severity of occurred Security-SLA violations.
The construction of the Security-SLA baseline that we used as a reference level when evaluating security could be discussed with more details in future work. Building an efficient and measurable Security-SLA is also a critical mission that is far from being complete. We are also planning to consider, along with security, other significant factors that could influence the Cloud federation formation process (e.g., costs, QoS), and study the trade-off that could exist between these factors, since defining utility functions that combine multiple aspects altogether is quite challenging.

References

[10] SPECS project (ICT-610795; Secure Provisioning of Cloud Services based on SLA Management): http://www.specs-project.eu/


Talal Halabi is a Ph.D. candidate in Computer Engineering at École Polytechnique de Montréal, Canada. He holds a Master of Research (MRes) degree in telecommunication and network engineering from the Lebanese University and Saint-Joseph University in Beirut. His current research activities are mainly focused on security quantification and evaluation in Cloud Computing, security satisfaction of Cloud consumers, Cloud Security-SLA development and standardization, optimal security-based Cloud resources allocation, and security of Cloud federations.

Martine Bellaiche is an Assistant Professor at the Computer and Software engineering Department of École Polytechnique de Montréal. She received a MSc degree in Computer Science from the University of Montreal in 1985 and her Ph.D. in Telecommunications from the National Institute of Scientific Research (INRS) in 2007. Her research is mainly focused on Network Security and attack detection, security in wireless sensor networks, and security of Cloud Computing and Internet of Things (IoT).