2018 2nd Cyber Security in Networking Conference (CSNet)

Cloud Security up for Auction: a DSIC Online Mechanism for Secure IaaS Resource Allocation

Talal Halabi, Martine Bellaiche, and Adel Abusitta
Department of Computer and Software Engineering
École Polytechnique de Montréal
Montréal, QC, CANADA
Email: {talal.halabi, martine.bellaiche, adel.abusitta}@polymtl.ca

***Abstract*—The lack of security is still one of the main factors that are blocking the full migration towards the Cloud Computing paradigm. More businesses would be attracted by the adoption of the Cloud model if the Cloud Infrastructure Providers (CIP) start increasing their investment in security solutions and demonstrating more explicitly the potency of their infrastructures in protecting customers' data and services. However, implementing security solutions is usually costly, and does not necessarily generate higher revenues. One way to reduce the cost of security investments would be to rent the Cloud secure resources to customers in a competitive fashion. The CIP could place her added value of security up for auction, and customers would place their bids along with their resource provisioning requests. In this paper, we propose an online mechanism that performs the allocation of the CIP's resources to customers in a security-oriented auction-based fashion. The mechanism is Dominant-Strategy Incentive-Compatible (DSIC), i.e., it guaranties the truthfulness of the bidders. The mechanism aims at allocating the resources to customers who valuate their security the most, and shows an acceptable performance compared to the famous offline Vickrey-Clarke-Groves (VCG) truthful mechanism.**

***Index Terms*—Cloud Computing, Cloud security, online mechanism, incentive-compatibility, truthful bidding.**

1. Introduction

The adoption of Cloud Computing technology continues to rise at an accelerated pace. Recent statistics [1] show that the total worldwide Cloud IT infrastructure revenue has almost tripled in the last four years. According to a new study [2], it is expected that in only 15 months, 80% of all IT budgets will be invested in Cloud applications and solutions. Many organizations are deciding to move their businesses to the Cloud to benefit from its attractive features such as scalability, resilience, and cost reduction, and deliver an enhanced user experience to their customers. In a typical Cloud Infrastructure-as-a-Service (IaaS) setting, a Cloud Infrastructure Provider (CIP) offers different types of resources (e.g., storage capacity, processing power, etc) to customers in the form of Virtual Machine (VM) instances, on which they deploy their applications and flexibly manage their workload. However, outsourcing data and services to a third party adds a new level of risk due to loss of physical control and introduces many security threats such as data breaches, data loss, and denial of service [3], which makes security an essential driving factor of the Cloud market today. Organizations expect from the CIP to maintain the security and availability of their data and services. Every year, the Cloud market giants witness severe incidents and security breaches causing Cloud outages and failures, which affect the production process and result in lost data and revenue. For instance, in 2017, Microsoft Skype Europe users suffered from connectivity problems due to an apparent Distributed Denial of Service attack (DDoS) that affected the whole communications platform [4]. Similarly, in 2016, a number of Amazon Web Services (AWS) VM instances hosting critical workloads for big companies subsequently failed because of a power outage in the region of Sydney, Australia, resulting in a serious service disruption [5]. Customers blamed AWS, the world's largest CIP, for not being sufficiently prepared for such incidents.

According to the statistics in [1], the rate of organizations that completely trust public Cloud infrastructures today to protect their data is only at 23%. The organizations usually tend to increase their trust in the security of public clouds and their use of the Cloud services when the level of integration with security solutions is high. However, investing in Cybersecurity by deploying advanced security solutions and providing a full security integration across multiple Cloud environments normally introduces additional costs to the CIP's budget, and does not necessarily generate higher profit. These costs include the price of security infrastructure (e.g., firewalls, antivirus software, intrusion detection systems, etc), salaries of security architects, and the expenses of security training programs. Investing in security is not necessarily a priority for the CIPs, but has apparently become indispensable to boost the migration towards the Cloud business model. The return on security investment in an IT infrastructure is not usually measured in terms of the achieved revenue, but as a function of damage reduction, i.e, the decrease in loss expectancy due to security incidents and the rate of threat occurrence. According to the study in [6], the average total cost of a data breach is around 4 million $. This gives an idea about how crucial is to secure a Cloud infrastructure that hosts thousands of services and huge amounts of users' sensitive data.

A way for the CIPs to reduce the cost of security investments would be to offer their security added value up for auction, and make sure that the customers who are renting their resources are the ones who valuate their security the most. A customer would include a bid in her resource provisioning request, which expresses her valuation of security. The bid does not involve the actual price of her requested VM package, which is estimated according to the CIP's fixed price scheme. Hence, the higher her bid is, the stronger her appreciation of the integrated security element is reflected. On the other hand, it will be possible for a customer to manipulate the auction and negatively affect the outcomes for the other bidders, by declaring untrue amounts of VMs in her request or a bid that is different from her actual valuation of the request. To

978-1-5386-7045-3/18/$31.00 ©2018 IEEE

avoid these situations, the auction should satisfy the incentive-compatibility property, i.e., to give incentives to bidders to reveal their true requests and valuations. In this paper, we design a Dominant-Strategy Incentive-Compatible (DSIC) online mechanism that allocates the CIP's secure resources to customers in real-time while guarantying their truthfulness, i.e., giving them incentives to reveal their actual resource provisioning requests and security valuations as a dominant strategy. The mechanism is based on a greedy allocation rule in which customers are prioritized according to their valuation of the security of these resources. Compared to the famous Vickrey–Clarke–Groves (VCG) [7] offline auction model, the proposed mechanism is computationally efficient and permits to achieve a good approximation of customers' optimal social welfare achieved by the allocation.

The remainder of the paper is organized as follows. Section 2 discusses the work that is related to the addressed topic. Section 3 describes the problem of allocation of Cloud IaaS secure resources, along with the mechanism design requirements. In Section 4, the proposed online mechanism is described. In Section 5, experimentation is performed and results are analyzed. Finally, section 6 concludes the paper.

2. Related Work

This work tackles the problem of resource allocation in the Cloud from a security perspective. Along with Quality of Service (QoS) and power efficiency, profit maximization has been an essential factor when it comes to resource allocation in the Cloud. For instance, Goudarzi and Pedram [8] proposed a distributed solution to a SLA-based resource allocation problem, which maximizes the total profit in the system while considering the following three dimensions in the optimization: processing, data storage, and communication bandwidth. Also, Nezarat and Dastghaibyfard [9] proposed a game theoretical model to maximize profit in a Cloud environment. In their model, a combinatorial auction mechanism is used to select the winners among the competent users.

The design of truthful resource allocation mechanisms in the Cloud has been previously studied by several researchers. For instance, Zaman and Grosu [10] proposed an online mechanism for VM provisioning and allocation in clouds. Their mechanism is incentive-compatible and aims at allocating VM instances whenever enough resources and matching bids are available. Zhang et al. [11] proposed an incentive-compatible online auction-based mechanism that allocates Cloud resources to users with heterogeneous demands. Their mechanism discourages the Cloud users from following a dishonest behavior when requesting resources, and its performance is comparable to that of the VCG mechanism. Nejad et al. [12] and Mashayekhy et al. [13] proposed an auction-based model for the problem of dynamic provisioning and allocation of VMs in the Cloud, and designed a truthful greedy mechanism that allocates the requests of the winning users and calculates their payments. Their mechanisms give incentives to users to be honest about their requests and valuations.

The main difference between these approaches and ours is that the latter aims at capturing customers' valuation of security, and not of the Cloud resources provided by the CIP. In our model, these resources will continue to be offered on the basis of a fixed-price scheme, but their security will be up for auction. If customers' rationality is assumed, i.e., their security valuations will be proportional to their need for security, the proposed mechanism will not only allow the CIP to reduce her cost of security investment, but also allocate her secure resources to the customers who need them the most. Eventually, the security risk level on her infrastructure and the probability of Cloud failures and data breaches will be reduced.

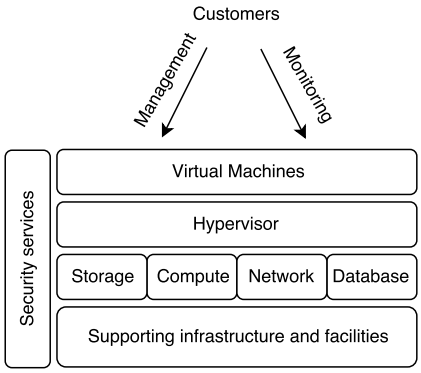


Fig. 1: The IaaS model basic architectural components.

3. System Model

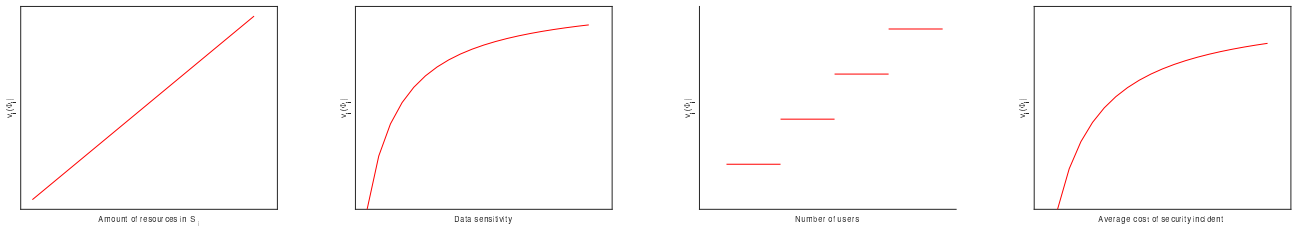
In this section, we present and model the problem of allocating the secure resources in a Cloud infrastructure in an auction-based context.

3.1. General Description

The Cloud IaaS delivery model consists in providing customers with a set of computational and storage resources in the form of VM instances which they will use to deploy their applications and perform their tasks. In a multi-tier architecture, these customers could be Cloud Service Providers (CSP) who rent these resources to provide different services (e.g., web hosting, scientific computing, financial simulations, email services, etc) to their users. Businesses normally tend to work with the IaaS model since it allows them to manage their workload without the need to manage the underlying infrastructure. Although customers could have the control over securing their VMs according to the security requirements of their applications, the process of securing the Cloud low level resources and supplying the customers with the right security capabilities is still in the hands of the CIP. The basic architecture of the IaaS model is depicted in Fig. 1. Securing the infrastructure would imply to secure every component of this architecture. For instance, implementing monitoring solutions on the hypervisor level, providing encrypted VM live migration, scanning and filtering VM images before creation, and configuring full VM backup capabilities could be all part of a security investment plan on the virtualization layer. Similarly, to ensure secure data storage, the CIP might need to implement robust encryption key management solutions, support secure data deletion, deploy data leakage prevention techniques, and increase the data backup frequency.

These security investment plans normally introduce additional costs to the CIP's budget, but are critical to protect her customers' workload and increase her trustworthiness. Customers would be ready to be part of this investment since they are the primary beneficent. In general, the CIP charges her customers on the basis of a pay per use service model according to the amount of resources they provision.

978-1-5386-7045-3/18/$31.00 ©2018 IEEE



(a) Security valuation in terms of the amount of resources requested by the customer.

(b) Security valuation with respect to the level of sensitivity or confidentiality of customer's data.

(c) Security valuation as a function of the number of users of customer's application.

(d) Possible relationship between customer's security valuation and the estimated cost of a security incident.

Fig. 2: Examples of customers' security valuation functions.

By keeping this model in place, and placing her resources up for auction after integrating the security aspect, the CIP will become aware of her customers' valuation of security. She will then be able to allocate her secure resources to those customers who valuate their security the most, and probably reduce her cost of security investment. This strategy will encourage the CIP to continuously improve the security level of her infrastructure, and allow the customers who are mostly in need for this increased level of security to benefit from it.

In our resource allocation model, we consider R types of resources provided by the CIP (e.g., number of CPU cores, memory amount, and storage capacity) and represent them in the set $\mathcal{R} = \{r,\ 1 \leq r \leq R\}$. We also consider K types of VMs that are offered by the CIP, each type provides a different combination of resources. The set of VM types is denoted by $\mathcal{VM} = \{k \mid 1 \leq k \leq K\}$. We consider a set $\mathcal{I} = \{i,\ 1 \leq i \leq I\}$ of customers, each requesting a specific VM package to manage their workload. We denote by $\Phi_i = (S_i, b_i)$ the request of customer $i \in \mathcal{I}$, where $S_i = (S_{i,1}, \ldots, S_{i,K})$ is a vector containing the required amounts of VM instances of each type $k \in \mathcal{VM}$, and b_i is the customer's bid, which is the monitory value that she is ready to pay to benefit from the added security element if her request is allocated to the infrastructure. The total usage $u_{i,r}$ of a resource $r \in \mathcal{R}$ by a request Φ_i is calculated as follows:

$$u_{i,r} = \sum_{k \in \mathcal{VM}} S_{i,k}\alpha_{k,r} \quad \forall i \in \mathcal{I},\ \forall r \in \mathcal{R} \tag{1}$$

where $\alpha_{k,r}$ is the usage of r by the VM type $k \in \mathcal{VM}$.

Customers in our model are assumed to be single-minded, meaning that, they are willing to pay their bids b_i if, and only if, they were able to get the allocation of their requests or a super set of them. The bid expresses the customer's valuation of security when her request is allocated for one normalized unit of time. Eventually, the payment that the customer will make will be computed using this value and according to the period of time for which her workload was executed, since this information is not always known prior to the allocation process. The bid b_i is generated by the customer's security valuation function defined as $v_i : \Phi_i \rightarrow \Re^+$, which reflects the benefit that customer i receives when her request Φ_i is allocated in the presence of security integration, with $v_i(\Phi_i) = b_i$. This benefit is usually related to the degree of damage that a security breach could cause to the customer's service or data, and multiple factors are normally involved in its estimation like the number of users of her application, the level of sensitivity or required confidentiality of their data, or the cost of service downtime.

In Fig. 2, four examples of customers' security valuation functions are shown. For instance, Fig. 2(a) illustrates how security valuation can be proportional to the amount of resources that the customer declares in her request, since provisioning more resources will most probably entail an increase in the application of security capabilities (e.g., provisioning more VMs would require to increase the number of storage discs available for backups). Fig. 2(b) highlights data sensitivity as the primary factor that helps the customer in evaluating her value of security. On the other hand, this valuation will eventually tend towards a limiting budget from the customer's perspective, which explains the concave increasing form of the function. Similarly in Fig. 2(c), the customer estimates her appreciation of the security element according to the number of users of her deployed Cloud application. When this number increases, the customer becomes responsible for protecting higher amounts of sensitive data that belong to the users, and will consequently be motivated to expand her investment in advanced security measures. This situation is also reflected in Fig. 2(d). Without loss of generality, the security valuation function is considered to be the customer's private information, and the allocation problem that we try to solve in this paper consists in effectively extracting this information from the customer during the auction.

3.2. Problem Formulation

In our model, we denote by $\widehat{\Phi_i} = (\hat{S}_i, \hat{b}_i)$ the declared request of customer $i \in \mathcal{I}$ and by $\Phi_i = (S_i, b_i)$ her true request. The problem consists in maximizing the social welfare [7] of the customers, which is a standard criterion that is used to evaluate the outcome of an auction mechanism. Maximizing the social welfare will allow the CIP to reduce her cost of security investments by allocating her secure resources to the customers who valuate them the most. Customers are assumed to be rational in our model, i.e., their security valuations will depend on how much they are in need for the secure Cloud resources. Hence, maximizing the social welfare will also allow the customers who mostly are in need for the security added value to benefit from it to protect their services and data. Eventually, the security risk level on the infrastructure will drop and the probability of security incidents will be reduced. The social welfare S is defined as the sum of security valuations of the allocated customers. In this security-oriented context, the secure Cloud resource allocation problem that aims at maximizing customers' social welfare can be defined as a Multi-dimensional Knapsack Problem (MKP) [15], which

978-1-5386-7045-3/18/$31.00 ©2018 IEEE

is a basic version in the class of Bin Packing problems. We formulate the problem as a Constrained Integer Linear Programming (CILP) problem as follows:

$$\text{Max} \quad S = \sum_{i \in \mathcal{I}} v_i(\widehat{\Phi}_i) x_i \tag{2}$$

Subject to:

$$\sum_{i \in \mathcal{I}} u_{i,r}\ x_i \leq \rho_r \quad \forall r \in \mathcal{R} \tag{3}$$

$$x_i = \{0, 1\} \quad \forall i \in \mathcal{I} \tag{4}$$

where x_i is a binary decision variable defined $\forall i \in \mathcal{I}$ as follows: $x_i = 1$ if Φ_i is allocated, and 0 otherwise. Constraint (4) guarantees that the allocation of all types of Cloud resources respects their available capacities $\rho_r,\ r \in \mathcal{R}$, with $\rho = (\rho_1, \ldots, \rho_R)$ is the vector of capacities provided by the CIP for each resource. The problem is considered to be strongly NP-hard [15], and finding the optimal solution is computationally inefficient for large scale problems. The truthful VCG auction mechanism can only work in an offline scenario and can not be deployed to dynamically handle customers' requests in a real-time fashion, since it requires the existence of an optimal allocation rule. Therefore, we design in this paper an online truthful mechanism to solve the problem of secure resource allocation in the Cloud.

3.3. Design Requirements

The aim of this paper is to design a Dominant Strategy Incentive-Compatible (DSIC) mechanism that allocates the CIP's secure resources to customers in real time, that is, a mechanism that gives incentives to customers to declare their true requests for resource provisioning and their true security valuations. Mechanism design [14] is an interesting approach to solving online allocation problems involving dynamic multi-agent environments where players should make truthful announcements for the sake of better system performance. Designing online mechanisms is becoming handy in multiple areas such as wireless networking, web servicing, and Cloud Computing. An online DSIC auction mechanism is usually defined by the combination of an allocation rule $\mathcal{A}(.)$ that determines which bidders receive their requested items, and a truth-inducing payment rule $\mathcal{P}(.)$ that determines based on the allocation results, the amount that each bidder must pay for her received item [7]. The allocation rule tries to maximize bidders' utility. The utility of a customer $i \in \mathcal{I}$ is defined as the difference between her valuation of her request and the payment P_i that she should make if the request is allocated:

$$utility_i = v_i(\widehat{\Phi}_i) - P_i \quad \forall i \in \mathcal{I} \tag{5}$$

Since bidders are assumed to be selfish players, their goal is to maximize their utilities, which could be done by manipulating their requests, that is, lying about the required amounts of resources or the placed bids. The objective of our mechanism is to prevent such manipulations by giving incentives to customers to reveal their true requests. Truthfulness in our case will ensure the right distribution of the secure Cloud resources to the customers who really need them. We denote by $\widehat{\Phi}$ the set of all declared requests and by $\widehat{\Phi}_{-i}$ the set of declarations excepted from the request of customer i. A mechanism is incentive-compatible if for every customer i, for every declaration of the other customers $\widehat{\Phi}_{-i}$, a true request Φ_i, and any other declared request $\widehat{\Phi}_i$ of customer i, we have $utility_i(\Phi_i, \widehat{\Phi}_{-i}) \geq utility_i(\widehat{\Phi}_i, \widehat{\Phi}_{-i})$. That means that declaring their true requests is a dominant strategy for the customers, which implies that they will maximize their utilities if they truthfully report their requests independently of the declarations of the other customers.

An incentive-compatible mechanism requires a monotone allocation rule, which means that if the rule allocates the request $\widehat{\Phi}_i$ to customer i as declared, then it will also allocate a more preferred request $\widehat{\Phi}_i^{'}$ of customer i. We define the preference relationship $\succeq$ as follows: a request $\widehat{\Phi}'_i = (\widehat{S'}_i, \hat{b}'_i)$ is more preferred than or equally preferred as a request $\widehat{\Phi}_i$ (denoted as $\widehat{\Phi}'_i \succeq \widehat{\Phi}_i$), if the amount of resources required to provision the vector $\hat{S}'_i$ is less than or equal to the amount required to provision $\hat{S}_i$, and $\hat{b}'_i \geq \hat{b}_i$. In other words, if the customer requests the provisioning of less resources and declares a higher security valuation in the request. The mechanism also requires the design of a payment rule that evaluates the payment of customers independently of their requests. The payment rule should be based on a critical value, which is a unique value b_i^c defined for each bidder i, such that all the requests that are more preferred than or equally preferred as (S_i, b_i^c) are winning declarations, and all the requests that are less preferred than (S_i, b_i^c) are losing declarations.

4. The Design of an Online Mechanism

In this section, we first apply the VCG mechanism to the problem of secure resource allocation in the IaaS Cloud. This mechanism is usually implemented in offline settings, where the auctioneer have all the information about future requests before the time of the allocation. Then, we describe the online truthful mechanism that we propose to respond to the dynamic and real-time requirements of our problem.

4.1. VCG-based Allocation

The VCG auction model [7] aligns all bidders' incentives with the goal of maximizing the social welfare, which is achieved by telling the truth. With a VCG-based mechanism, the allocation rule is usually required to be implemented using an optimal allocation algorithm. In our case, we integrate the CILP problem presented in the previous section in the design of a VCG-based mechanism to find the optimal allocation. The mechanism also requires the implementation of a payment rule that computes the payments of customers based on the allocation results, which is defined as follows:

$$P_i = \sum_{j \in \mathcal{A}(\widehat{\Phi}_{-i})} v_j(\widehat{\Phi}_j) - \sum_{j \in \mathcal{A}(\widehat{\Phi}), j \neq i} v_j(\widehat{\Phi}_j) \quad \forall i \in \mathcal{I} \tag{6}$$

where the first term of the equation reflects the optimal social welfare that would have been achieved by all players if customer $i \in \mathcal{I}$ had not participated in the auction (allocation of $\widehat{\Phi}_{-i}$), and the second term represents the social welfare achieved by all players except for i, when i had participated in the auction (allocation of $\widehat{\Phi}$). In other words, the payment of a bidder is evaluated as being the harm that the bidder had caused to the other bidders by participating in the auction.

The VCG-based allocation mechanism is presented in Algorithm 1. It consists of performing the optimal allocation by solving the CILP problem (Eqs. (2), (3), and (4)) that

978-1-5386-7045-3/18/$31.00 ©2018 IEEE

Algorithm 1 Offline VCG-based allocation.

Input:
- Customers' declared requests in the set Φ
- The vector of resource capacities ρ

Output:
- The allocation vector x
- The payment vector P

1: (x, S) = Solve the CILP problem for $\widehat{\Phi}$
2: **for all** $i \in \mathcal{I}$ **do**
3: **if** $x_i = 1$ **then**
4: (x^*, S^*) = Solve the CILP problem for $\widehat{\Phi}_{-i}$
5: $P_i = S^* - (S - \hat{b}_i)$
6: **else**
7: $P_i = 0$
8: **end if**
9: **end for**
10: $P \leftarrow (P_1, \ldots, P_I)$
11: **return** x, P

maximizes the social welfare, and implementing the payment rule defined by the VCG model (Eq. (6)). The mechanism takes two inputs: the set of customers' declared requests $\widehat{\Phi}$ and the vector of Cloud resource capacities ρ, and generates two outputs: the optimal allocation vector x, and the vector of payments P. First, the mechanism determines the optimal allocation of requests (line 1) using a dynamic programming optimization algorithm and generates the allocation vector x and the optimal social welfare S. Then, it computes the payment of each customer (lines 2-9). According to the VCG model, to determine the payment of an allocated customer $i \in \mathcal{I}$, we solve the allocation optimization problem without the participation of i (line 4) and compute the social welfare S^* achieved by the new allocation vector x^*. The payment of the customer will then be obtained by computing the difference between the two values S^* and $S - \hat{b}_i$ (line 5). If this difference is null (the requests allocated by x and x^* are exactly the same), or customer's request was not allocated, the customer will have nothing to pay. In other words, the participation of the customer in the auction had no affect on the other players.

The VCG-based mechanism is incentive-compatible [7], and provides the CIP with the optimal resource allocation strategy. However, the defined CILP problem is strongly NP-hard and becomes computationally inefficient when the number of customers and the amounts of available resource capacities become large, which makes the mechanism suitable to be executed only in an offline setting. To be able to allocate the Cloud resources on the fly while guarantying the truthfulness of customers, we need to design an incentive-compatible allocation mechanism that is able to provide a good approximation of the optimal social welfare in polynomial time.

4.2. The Proposed Mechanism

We design in this section an incentive-compatible mechanism that allocates the secure Cloud resources in real-time according to customers' security valuations. In the case where large instances of the problem are handled (e.g., tens of thousands of items offered for sale), finding the optimal allocation becomes computationally exhausting. The goal here is to design a non-VCG mechanism that finds an acceptable level of approximation of the optimal solution using heuristics, in an incentive-compatible fashion. The mechanism should optimize the social welfare as much as possible, while guarantying the truthfulness of the bidders. In other words, it should give incentives to customers to always prefer to truthfully report their private valuations of the offered security settings to the CIP, rather than any malicious or dishonest declaration that could have a potential in increasing their utilities.

4.21. Mechanism Description

Since customers provision the requested VM instances in the form of R types of Cloud resources, any request $\hat{\Phi}_i$ with a vector $\widehat{S}_i$ of requested VMs can be mapped into an R-dimensional space of items, where the smallest item in this space contains one unit of each of the normalized R resources allocated for one unit of time. The goal here is to allocate these items according to the valuations of their security.

The online mechanism is presented in Algorithm 2, and is executed in the event where resource provisioning requests were placed at a time $t \in T$ (T is the infinite interval of time). Before the execution, the amount of remaining resources at time t within the Cloud is computed by considering all the previously allocated requests which execution is not yet completed. We denote by $\rho^t = (\rho_1^t, \ldots, \rho_R^t)$ the vector of available capacities of resources of all types at time t. The mechanism involves two stages: an allocation procedure and payment calculation. It receives two inputs: the vector of available resource capacities ρ^t at time t, and the set $\widehat{\Phi}^t = \{\widehat{\Phi}_i \mid i \in \mathcal{I}, \widehat{\Phi}_i$ is declared at time $t\}$ of requests, and generates three outputs: the allocation vector x, the payment vector P, and the updated vector of available resource capacities ρ^{t+1}. The mechanism starts the execution only if there were new requests that were placed at time t, and if resources are available within the Cloud (line 1). In this case, the mechanism will perform a heuristic allocation of requests using a greedy algorithm [7] and find the allocation vector x (lines 3-11). Next, the mechanism will compute the vector P of payments that should be made by the allocated customers (lines 14-34).

4.22. Allocation Procedure

Let $\hat{u}_{i,r}$ be the requested amount of resource $r \in \mathcal{R}$ by customer $i \in \mathcal{I}$ computed using Eq. (1) according to the vector $\hat{S}_i$, and let $\hat{u}_i$ be the vector containing the declared amounts of all resource types. In our mechanism, the allocation is performed using a greedy procedure and based on the Valuation of Security per Item (VSI) parameter that we define for a bidder $i \in \mathcal{I}$ as follows:

$$VSI_i = \frac{\hat{b}_i}{\prod\limits_{r \in \mathcal{R}} \hat{u}_{i,r}} \tag{7}$$

The VSI_i parameter reflects how much customer i valuates the security of each unit of resources allocated for one unit of time. In other words, it shows the customer's security valuation for the smallest item in the R-dimensional space of items.

The mechanism allocates the Cloud resources to customers in decreasing order of their declared VSI. It first sorts the requests $\widehat{\Phi}_i$ placed at time $t \in T$ in $\widehat{\Phi}^t$ in decreasing order of their VSI_i (line 5), then performs the allocation of the sorted requests while resources are still available in ρ^t, and updates the allocation vector x (lines 6-11). The mechanism adopts this concept since it aims at maximizing the social

978-1-5386-7045-3/18/$31.00 ©2018 IEEE

Algorithm 2 The proposed online mechanism.

Input:
- The vector of available resource capacities ρ^t
- The set of requests $\widehat{\Phi}^t$

Output:
- The allocation vector x
- The payment vector P
- The updated vector of resource capacities ρ^{t+1}

1: **if** $\hat{\Phi}^t \neq \emptyset$ and $\rho^t \neq 0$ **then**
2: {Heuristic allocation of ρ^t to $\widehat{\Phi}^t$}
3: $x \leftarrow$ null vector of size I
4: Calculate $VSI_i \ \forall \widehat{\Phi}_i \in \widehat{\Phi}^t$
5: Sort $\widehat{\Phi}_i \in \widehat{\Phi}^t$ in decreasing order of VSI_i
6: **for all** $\widehat{\Phi}_i \in \widehat{\Phi}^t$ in decreasing order of VSI_i **do**
7: **if** $\rho_r^t - \hat{u}_{i,r} \geq 0 \ \ \forall r \in \mathcal{R}$ **then**
8: $\rho^t \leftarrow \rho^t - \hat{u}_i$
9: $x_i \leftarrow 1$
10: **end if**
11: **end for**
12: $\rho^{t+1} \leftarrow \rho^t$
13: {Payment}
14: **for all** $\widehat{\Phi}_i \in \widehat{\Phi}^t$ in decreasing order of VSI_i **do**
15: **if** $x_i = 1$ **then**
16: $\widetilde{\rho} \leftarrow \rho^t$
17: $\widetilde{\rho} = \widetilde{\rho} + \hat{u}_i$
18: {Allocation of $\widetilde{\rho}$to $\widehat{\Phi}^t \backslash \widehat{\Phi}_i$}
19: $found \leftarrow FALSE$
20: **for all** $\widehat{\Phi}_j \in \widehat{\Phi}^t \backslash \widehat{\Phi}_i$ in decreasing order of VSI_j and with $VSI_j < VSI_i$ **do**
21: **if** $x_j = 0$ and $\widetilde{\rho}_r - \hat{u}_{j,r} \geq 0 \ \ \forall r \in \mathcal{R}$ **then**
22: $P_i \leftarrow VSI_j \ . \ \prod_{r \in \mathcal{R}} \hat{u}_{i,r}$
23: $found \leftarrow TRUE$
24: break
25: **end if**
26: **end for**
27: **if** $found$ is $FALSE$ **then**
28: $P_i \leftarrow 0$
29: **end if**
30: **else**
31: $P_i \leftarrow 0$
32: **end if**
33: **end for**
34: $P \leftarrow (P_1, \ldots, P_N)$
35: **return** x, P, ρ^{t+1}
36: **else**
37: **return**
38: **end if**

welfare of the players, which is the sum of the reported bids by the allocated customers. The mechanism helps the CIP in reducing the cost of security investments by allocating the secure resources to the customers who are ready to pay more for their security. Moreover, it aims at providing the secure resources to customers who really need them. The allocation procedure tries to approximate the optimal social welfare in polynomial time. Sorting customers' requests introduces a time complexity of $O(n \ log \ n)$ on average, where n is the number of requests, and verifying their allocation feasibility requires $O(n)$. Hence, the overall time complexity of the proposed allocation procedure is $O(n(log \ n + 1))$.

4.23. Payment Calculation

After the mechanism finds the near-optimal allocation of resources at time t, it computes the payment of each allocated customer, which is considered to be the minimum value a customer should declare in order to get the requested secure resources. To compute the payment of an allocated customer at time t, the mechanism implements the concept of Eq. (6) in real-time. First, it supposes that customer $i \in \mathcal{I}$ was not participating in the auction, and re-adds the resources that were allocated to i to the vector ρ^t by creating a new vector $\tilde{\rho}$ (lines 16-17). Then, it starts allocating the resources in $\tilde{\rho}$ to the customers that have lower VSI values than that of i in the set $\widehat{\Phi}^t \backslash \widehat{\Phi}_i$, and which were not allocated the first time, when customer i was included. The mechanism stops when the first customer j who has the VSI value lower than VSI_i and who was not allocated when customer i was participating in the auction, gets allocated when customer i did not participate (lines 19-26). The mechanism evaluates the payment of customer i based on the security valuation VSI_j of customer j (line 22), which constitutes the highest VSI value among losing bidders. The intuition behind this is that bidder j is the bidder who lost exactly due to the participation of bidder i in the auction. This payment is the minimum value that the customer should report to get her request. If no such customer was found, or the request of customer i was not allocated by the mechanism, i will be required nothing to pay (lines 27-32). In other words, her participation in the auction did not affect the other participating customers, which would have achieved the same social welfare, if customer i had not participated.

4.24. Incentive-Compatibility

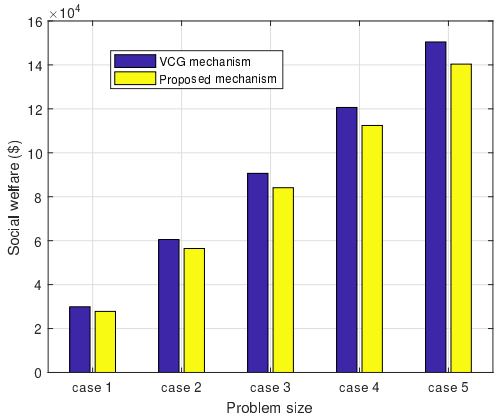
The proposed mechanism is incentive-compatible, since it satisfies the monotonicity and critical value properties. For instance, if a bidder $i \in \mathcal{I}$ is able to allocate her request $\widehat{\Phi}_i$, she will also be able to allocate a more preferred request $\widehat{\Phi}_i' \succeq \widehat{\Phi}_i$, since declaring a vector $\hat{S}_i'$ with less required resources than in $\hat{S}_i$, or a bid $\hat{b}_i' \geq \hat{b}_i$ will only help the bidder in increasing her VSI_i and moving up in the greedy order, making it easier to win. This implies that the bidder will not have incentives to declare a different request than the one she actually wants in order to guarantee her winning.

The mechanism implements payment calculation based on the minimum value that the bidder must declare in order to get the allocation of her request. Hence, if bidder i declares a lower value than this minimum value, she will lose the allocation; otherwise she will win. This minimum value is the critical value of bidder i. Since our mechanism implements a monotone allocation procedure and a payment calculation based on the critical value, it is considered incentive-compatible [7].

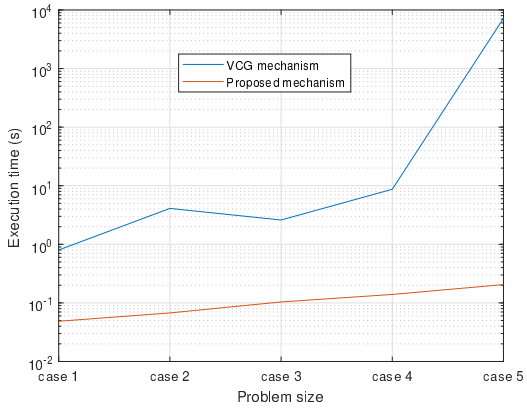
5. Experimentation and Results

In this section, we conduct a set of evaluation experiments and analyze the results. We implement the VCG-based allocation mechanism and the proposed online mechanism in MATLAB, and use a Mixed-Integer Linear Programming solver to find the optimal solution to the CILP problem defined in Eqs. (2), (3), and (4). The solver first tries to reduce the problem size, then uses heuristics to solve an initial relaxed problem and produce an initial feasible solution, and finally applies a Branch and Bound algorithm [16] to perform an

978-1-5386-7045-3/18/$31.00 ©2018 IEEE



(a) The social welfare achieved by the two mechanisms.



(b) The execution time of the two mechanisms.

Fig. 3: Evaluation of the performance of the online mechanism with respect to the offline VCG mechanism in five different scenarios: case 1, $I = \rho_r = 5000,\ \forall r \in \mathcal{R}$; case 2, $I = \rho_r = 10000$; case 3, $I = \rho_r = 15000$; case 4, $I = \rho_r = 20000$; case 5, $I = \rho_r = 25000$.

Table 1: VM types offered by Amazon EC2.

	Small (k=1)	Medium (k=2)	Large (k=3)	ExtraLarge (k=4)
CPU cores (r=1)	1	2	4	8
Memory (GB) (r=2)	1.7	3.75	7.5	15
Storage (GB) (r=3)	160	410	850	1690

exhaustive search for the optimal solution. The experiments are performed on a 64-bit Windows 7 machine equipped with an Intel Core i7-3612QM CPU @ 2:10 GHz Processor and 12 GB RAM.

5.1. Experimental Setup

The experiments aim at evaluating the performance of the designed mechanism in terms of the approximation of the social welfare and computational efficiency, as well as demonstrating the incentive-compatibility property. We consider the three standard types of Cloud resources in our experimentation, the number of CPU cores, the amount of memory (GB), and storage capacity (GB), and four different types of offered VM instances like the ones offered by Amazon EC2. The four types are presented in Table 1 along with their parameters $\alpha_{k,r}$ which correspond to the usage of resource type $r \in \mathcal{R}$ by the VM type $k \in \mathcal{VM}$. For each request, the types of VMs are randomly selected and a random number between 0 and 10 is generated to simulate the number of requested VMs of each type. Finally, we randomly generate a value between 1 and 20$ to represent the declared security valuation (bid) in the request.

5.2. Results Analysis

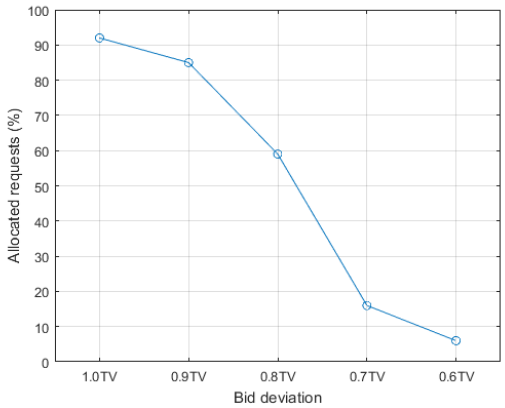
To study the performance of the mechanism, we consider multiple scenarios where we vary the allocation problem size by increasing the number of requests and the amounts of available capacities of Cloud resources. Fig. 3 shows the results obtained for different combinations of I and ρ_r, $r \in \mathcal{R}$. In Fig. 3(a), the social welfare achieved by the two mechanisms in each case is presented. The results shown are the average of fifty runs. We notice that the social welfare obtained by the proposed mechanism is very close to the optimal one achieved by the VCG mechanism. The greedy algorithm usually achieves a good approximation of the optimal solution (to at least 50%) for some specific problems, such as Knapsack problems.

In Fig. 3(b), we show the execution time of the two mechanisms on a logarithmic scale since the values for some of the cases are very distanced. As shown, the VCG-based allocation is relatively slow and becomes computationally inefficient in the case of large scale problem instances since it aims at finding the optimal allocation rule, which makes it suitable to perform only in an offline allocation scenario. On the other hand, the greedy-based allocation mechanism is very fast and is able to find a near-optimal solution in polynomial time, which makes it ideal to be implemented in a real-time mode.

To study the incentive-compatibility of the designed mechanism, we consider a total of a hundred cases where a customer $i \in \mathcal{I}$ participates in the auction while varying her reported bid. In Fig. 4(a), we show the variation in the request's allocation rate along with the rate of negative utility when the bid value positively deviates from the true valuation of the request, that is, when the customer reports a higher value than her true security valuation in the request with the objective of increasing her chances of getting her request allocated. We denote by TV the true valuation of customer's request. We consider a case where $TV = 10\$$ and the vector of resources requested by the customer is equal to (2,3,4). We notice that when customer i increases her bid by reporting a value $\hat{b}_i$ higher than her true valuation, her chances of getting her request allocated will increase, which is perfectly normal since reporting a higher bid implies a higher value of computed VSI_i, which will move the customer up in the greedy algorithm and increase her chances of getting allocated by the mechanism. On the other hand, this strategy comes with a cost, which is increasing the possibility of achieving a negative utility. We use the achieved utility metric to demonstrate the incentive-compatibility of the mechanism. For instance, when the reported bid was equal to $1.1TV$, the customer achieved an allocation rate of 20%, but in 75% of the cases where the customer got her request allocated, her utility, which is equal to the difference between her true security valuation and her made payment, was negative. This goes back to the fact that customer's payment is computed based on the bids of the other customers, and in some cases, it could be higher than the actual valuation of the customer. This proves that the mechanism guarantees the truthfulness of the customers and gives them incentives to report their true requests, since deviating from the truth will not always achieve a positive utility, hence it is not an optimal strategy.

978-1-5386-7045-3/18/$31.00 ©2018 IEEE

(a) Request allocation and negative utility rates when declared bids are higher than the true security valuation.



(b) Allocation rates of customer's request when declared bids are lower than her true security valuation.

Fig. 4: Study of incentive-compatibility of the proposed mechanism based on the bid deviation from the true security valuation.

In Fig. 4(b), we show the variation in the allocation rate of customer's request when the reported bid value negatively deviates from the true valuation. We start by defining a setting where the customer is able to obtain the allocation of her request in most of the hundred cases. The value of TV is set to 10$ and the vector of requested resources to (2,2,3). When decreasing the reported bid and performing the allocation, the allocation rate starts to drop, hence following this strategy by the customer will not always guarantee the allocation of her request. We can conclude that the designed mechanism satisfies the incentive-compatibility property, where telling the truth is a dominant strategy for the customers.

6. Conclusion

The lack of security and trustworthiness in Cloud Computing is still slowing down its adoption, especially by the organizations and businesses that deal with sensitive data. CIPs are required to increase their investments in security solutions to boost the migration towards the Cloud. However, investing in cybersecurity adds considerable costs to their budgets, and does not necessarily generate higher profit. We proposed in this paper an online mechanism that performs the allocation of CIPs' secure resources in an auction-based fashion. In our allocation model, customers would be required to show their appreciation of the security added value by including bids in their resource provisioning requests. The mechanism implements an online allocation rule that places customers' requests according to their security valuation, and a truth-inducing payment rule that computes their required payments. This idea allows the CIP to reduce the cost of security investments, allocate their secure resources to the customers who are mostly in need for protection, and be encouraged to increase her investments in the future. Our mechanism showed acceptable performance compared to the offline VCG mechanism. It achieved acceptable approximation of the optimal social welfare and is implementable in real-time. In the future, we plan on identifying the security resources that could be placed for auction and integrate them into the model.

References

[1] Cloud Tweaks, Driven by expansion in public Cloud. October 6, 2017. Online: https://cloudtweaks.com/2017/10/worldwide-cloud-infrastructure-revenues-grow-25-8-second-quarter-2017/ (accessed on February 15, 2018).

[2] Forbes, 2017 State Of Cloud Adoption And Security. April 23, 2017. Online: https://www.forbes.com/sites/louiscolumbus/2017/04/23/2017-state-of-cloud-adoption-and-security/#5788f24f1848 (accessed on February 15, 2018).

[3] Los, R., Shackleford, D., & Sullivan, B. (2013). The notorious nine cloud computing top threats in 2013. Cloud Security Alliance.

[4] Joseph Tsidulko, The 10 Biggest Cloud Outages of 2017. August 1, 2017. Online: http://www.crn.com/slide-shows/cloud/300089786/the-10-biggest-cloud-outages-of-2017-so-far.htm/pgno/0/9 (accessed on February 21, 2018).

[5] Joseph Tsidulko, The 10 Biggest Cloud Outages of 2016. December 29, 2016. Online: https://www.crn.com/slide-shows/cloud/300083247/the-10-biggest-cloud-outages-of-2016.htm (accessed on February 21, 2018).

[6] Ponemon Institute LLC, 2016 Cost of Data Breach Study: Global Analysis. (June 2016).

[7] Nisan, N., Roughgarden, T., Tardos, E., & Vazirani, V. V. (Eds.). (2007). Algorithmic game theory (Vol. 1). Cambridge: Cambridge University Press.

[8] Goudarzi, H., & Pedram, M. (2011, June). Maximizing profit in cloud computing system via resource allocation. In Distributed Computing Systems Workshops (ICDCSW), 2011 31st International Conference on (pp. 1-6). IEEE.

[9] Nezarat, A., & Dastghaibyfard, G. (2016). A game theoretical model for profit maximization resource allocation in cloud environment with budget and deadline constraints. The Journal of Supercomputing, 72(12), 4737-4770.

[10] Zaman, S., & Grosu, D. (2012, June). An online mechanism for dynamic vm provisioning and allocation in clouds. In Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on (pp. 253-260). IEEE.

[11] Zhang, H., Jiang, H., Li, B., Liu, F., Vasilakos, A. V., & Liu, J. (2016). A framework for truthful online auctions in cloud computing with heterogeneous user demands. IEEE Transactions on Computers, 65(3), 805-818.

[12] Nejad, M. M., Mashayekhy, L., & Grosu, D. (2015). Truthful greedy mechanisms for dynamic virtual machine provisioning and allocation in clouds. IEEE transactions on parallel and distributed systems, 26(2), 594-603.

[13] Mashayekhy, L., Nejad, M. M., Grosu, D., & Vasilakos, A. V. (2016). An online mechanism for resource allocation and pricing in clouds. IEEE transactions on computers, 65(4), 1172-1184.

[14] Parkes, David C. Online mechanisms. (2007).

[15] Kellerer, H., Pferschy, U., & Pisinger, D. (2003). Knapsack problems. 2004.

[16] Clausen, J. (1999). Branch and bound algorithms-principles and examples. Department of Computer Science, University of Copenhagen, 1-30.

978-1-5386-7045-3/18/$31.00 ©2018 IEEE