The Journal of Systems & Software 187 (2022) 111215

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

A sampling-based online Co-Location-Resistant Virtual Machine placement strategy☆

Marwa Thabet [a,c,*], Brahim Hnich [b,c], Mouhebeddine Berrima [d,e]

[a] University of Sousse, ISITCom, 4011, Sousse, Tunisia
[b] Faculty of Sciences, University of Monastir, Monastir, Tunisia
[c] CES-Laboratory, University of Sfax, Sfax, Tunisia
[d] Department of Applied Natural Sciences, Applied College, Qassim University, Saudi Arabia
[e] LIPSIC, University of Tunis El Manar, Tunisia

ARTICLE INFO

Article history:
Received 4 June 2021
Received in revised form 22 November 2021
Accepted 3 January 2022
Available online 11 January 2022

Keywords:
Cloud security
Resource optimization
Co-location attack
Virtual machines placement
Co-Location-Resistant assignment
Sampling

ABSTRACT

In this paper, we discuss the co-location attack problem in the cloud IaaS from the Virtual Machine (VM) placement strategy perspective. We formulate the online secure optimization VM placement problem in a way that guarantees—apriori—a specified level of security while minimizing the number of used physical servers. To solve such a problem, we propose an approximate online secure VM placement algorithm based on sampling. The polynomial-time algorithm is based on a sound security inference procedure based on the confidence interval estimation method. Our empirical results demonstrate the correctness and the effectiveness of our approach in guaranteeing a Co-Location-Resistant (CLR) VM placement with a specific level of confidence and a threshold error as new incoming VM requests are being assigned to servers online. We compared our algorithm to a CLR alternative presented in Azar et al. (2014).

1. Introduction

Cloud security threats. Cloud computing emerges as a multi-disciplinary technology. It, notably, affects business, innovation, education and marketing. Cloud services are distributed and scalable where different users can share several resources, processing and data. As soon as they need, users known also as the cloud customers, can consume the required resources as services and pay for usage. This enables them to save money and get rid of hardware and software management (Mell and Grance, 2011). The multi-tenancy concept is crucial in a cloud setting to maintain a high resource utilization, cost savings for customers and high revenues for cloud providers. Cloud computing includes several multi-tenant service models e.g., Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), Software-as-aService (SaaS) and Containers-as-a-Service (CaaS). At the IaaS level, multi-tenancy enables data center's hardware to get virtualized, which allows cloud providers to run multiple Virtual Machine (VM) instances on a single physical machine (Chowdhury et al., 2015). However, despite the isolation between tenants ensured by the cloud hypervisor, multi-tenancy is subject to many security concerns. The studies in Ristenpart et al. (2009), Xu et al. (2015) and Varadarajan et al. (2015) affirm that many security properties like confidentiality, integrity and availability can be compromised due to the emerging attacks such as side channel (Ristenpart et al., 2009; Zhang et al., 2014), covert channel (Wu et al., 2012) and power attacks (Xu et al., 2014). Side and covert channels aim at extracting sensitive information (e.g., cryptographic keys), or information (e.g., cache utilization, workload and web traffic) to plan for further attacks. Basically, to achieve the above attacks, the first step is to co-locate some VMs of an attacker with at least one target (or victim) VM on the same physical server. These co-location (or co-residence) attacks are one of the most challenging security threats, and have received research attention in different service models such as IaaS (Gupta et al., 2020; Ristenpart et al., 2009), PaaS (Zhang et al., 2016), SaaS (Levitin et al., 2020, 2021) and CaaS (Shringarputale et al., 2020).

Co-location attacks. The first academic work evoking the problem of co-location attacks was published in Ristenpart et al. (2009). In this work, the author showed that such an attack is successful if the attacker co-locates with a good fraction of target VMs known as attack coverage and, that is easy to do on Amazon EC2, by employing two strategies for launching (typically a large number of) attacker VMs. The first is the *parallel placement strategy*, in which the attacker VMs are launched simultaneously or within

☆ Editor: J.C. Duenas.

* Corresponding author at: University of Sousse, ISITCom, 4011, Sousse, Tunisia.
E-mail addresses: marwa.thabet@isitc.u-sousse.tn (M. Thabet), brahim.hnich@fsm.rnu.tn (B. Hnich), berrima.mouheb@gmail.com (M. Berrima).

https://doi.org/10.1016/j.jss.2022.111215
0164-1212/

a short time window with the target VMs. This strategy exploits the observation affirming that the VMs launched simultaneously or within a short time window are often co-located. The second is *sequential placement strategy*, in which a large number of attacker VMs are launched over a long period to be scattered on the most physical servers, and consequently, increases the likelihood of co-location. Through these ways, the adversary influences the way VMs are injected to the algorithm that schedules them and hence may control the placement strategy. Few years later, two other studies were conducted to analyze co-location attacks on public clouds. In Xu et al. (2015) the authors showed that the co-location threats exist on Amazon EC2 with a slight mitigation. The study in Varadarajan et al. (2015) demonstrated that the three public clouds Amazon EC2, Microsoft Azure and Google Compute Engine are vulnerable to co-location attacks. Recently, Gupta et al. (2020) replicated the co-residency test of Ristenpart et al. (2009) on AWS and OpenStack, and concluded that co-location threats remain a real issue for more than a decade after their demonstration. In the literature, there is a common interest in mitigation techniques, which are suitable to immediate deployment, i.e., this does not require any changes to the operating system, the hardware or applications, such as techniques based on data replication (Levitin and Xing, 2021) and security-aware VM placement strategies, e.g Hasan and Rahman (2020), Rethishkumar and Vijayakumar (2019), Xiao et al. (2021) and Long and Duong (2020).

VM placement strategy. A typical VM placement strategy focuses on resource optimization. It assigns servers to incoming VMs so that the selected server provides the required resources by the VMs and satisfies some resources constraints. Generally, the constraints are expressed by minimization or maximization functions of some resources like energy consumption (Mann, 2015; Pires and Barán, 2015), network traffic (Mann, 2015; Mills, 2013) and utilization rate. In the data centers, energy consumption is the most expensive resource, which is correlated with the number of the machines switched on. Thus, optimizing energy consumption is reduced to minimizing the number of active physical machines (Azar et al., 2014; Mann, 2015). When minimizing the number of physical machines, the VM placement problem is an instance of the online bin packing problem (BPP), which is a well known NP-hard problem (Seiden, 2002). The BPP seeks to pack bins, within an online setting, with a given weight into the minimum number of bins with respect to the capacity constraints. It is modeled as a mathematical problem and solved using Integer Linear Programming (ILP) (Ferreto et al., 2011). Obviously, such exact methods cannot respond quickly to a large number of instances. Thereby, VM placement problem exploits the classical heuristics designed for the online BPP problem. First Fit and Best Fit (Bobroff et al., 2007) guarantee the best solution when VM size and servers' capacity are unified. Nevertheless, these heuristics may lead to overloaded hosts. To balance the workload, there are heuristics such as Worst Fit (Fang et al., 2013) that chooses the server with the maximum amount of available resources that fits the VM.

Problem statement & contributions. The deterministic nature of VM placement strategies enables attackers to predict their placement. An adversary can make some observations, which can be exploited to increase the probability of co-location (Ristenpart et al., 2009). To mitigate co-location attacks, a naive strategy, namely the simple random strategy, assigns randomly VMs to servers. In this way, it disables users, specially malicious ones, from guessing the VM placement. Hence, it minimizes the probability of co-location attacks, but at the cost of a significant overhead in terms of resources consumption. Therefore, it cannot be applied in practice. Furthermore, the possibility of launching VMs from different accounts makes it difficult to prevent co-location attacks by identifying the attacker VMs.

Our proposed statistical approach for the online secure optimization VM placement problem is inspired by the work of Azar et al. (2014). We introduce the notion of a co-location resistant assignment of a VM to a server. Based on sampling, we design a security inference procedure, which evaluates if an assignment is co-location resistant or not with a certain level of confidence and threshold error level. This procedure is used by our VM placement strategy to guarantee apriori a given co-location attack probability threshold. To optimize the resources, we extend the Best Fit algorithm and use it to assign the VMs to the servers. Compared to the existing works, the advantages and features of our secure VM placement strategy are: (i) it models the uncertainties related to VM types (attack, target and neutral) in a probabilistic manner, (ii) it guarantees *a-priori* co-location attack probability through enforcing novel probabilistic security constraints, and (iii) the co-location resistance is ensured throughout the placement of all the VMs. At a high level, ensuring the co-location resistance apriori is very useful since it allows establishing a security level agreement between the cloud provider and the customer. We validate and evaluate our approach through exhaustive simulations of different cloud configurations. Our empirical results show that our approach is sound and is able to effectively place incoming VMs on servers while guaranteeing apriori a certain level of co-location resistance.

The rest of the paper is organized as follows. In Section 2, we review the related works addressing the secure VM placement strategies. Next, in Section 3, we describe our novel definition of the secure optimization virtual machines placement problem in the cloud. In Section 4, we present the theoretical foundations necessary for our new approximate sampling-based CLR VM placement strategy given in Section 5. In Section 6, we validate our statistical inference method while, in Section 7, we exhaustively evaluate its performance. Finally, before concluding the paper in Section 9, we show how to deploy the proposed method in practice and assess the trade-off between security and resource optimization.

2. Related works

The problem of secure optimization VM placement algorithms was introduced rigorously in Azar et al. (2014). The authors exposed the first formal building blocks of these algorithms tackling co-location attacks while optimizing resources. They formalized both notions of security and efficiency of a VM placement algorithm. Inspired from theoretical cryptography, the security was expressed as a notion of co-location resistance. Intuitively, given n servers, q attacker VMs, t target VMs, a fractional number $\delta \in]0, 1]$, and a small number ϵ; a placement algorithm is $(n, q, t, \delta, \epsilon)$-Co-Location-Resistant if the adversary's success probability of co-locating with at least $\delta \times t$ target VMs is at most ϵ. The parameter ϵ refers to the co-location attack probability upper bound (threshold). According to δ, the co-location attacks are split into three types: (i) *complete* attacks when $\delta = 1$, i.e. each target VM is co-located with at least one attacker VM, (ii) *fractional* attacks when $0 < \delta < 1$, i.e. some target VMs are co-located with at least one attacker VMs, and (iii) otherwise a *single* attack when $\delta = 1/t$. Taking into account efficiency, the authors defined a new cost function returning the cumulative number of used servers over time.

The proposed algorithm in Azar et al. (2014), abbreviated by CLR, is an intuitive extension of the random strategy parameterized by λ –a subset of "open" servers allowed to host VMs. The parameter λ allows us to balance between security and optimization; the larger λ is, the smaller the probability of co-location is, but the worse the optimization is. Indeed, when λ is equal to the total number of servers, CLR coincides with the random

strategy. Initially, λ servers are open and the rest of VMs are closed. Whenever a VM request arrives, the algorithm proceeds as follows: it assigns the VM to a server selected uniformly at random from the open servers. If this server becomes full, it is removed from the set of open servers and replaced randomly with a closed one. The main result states that if $t \leq \lambda/e$ and $q < \lambda.\log(t/2)$ then the proposed algorithm CLR is a complete co-location resistant placement strategy ($\delta = 1$) for a certain value ϵ in function of t, q and some constant $\gamma < 1$. A related grouping-based VM placement strategy (SR-FF) was proposed in Liang et al. (2017). Initially, the servers are divided into λ groups. Then, given a VM request and the current state of servers (load and capacity), the placement of this VM is performed by selecting randomly a group among the λ groups and then selecting a server from the selected group with respect to an optimization strategy (such as the First Fit heuristic). The random group selection prohibits the prediction of VMs placement while the host selection allows to optimize the number of running servers. Theoretical and simulation results affirm that SR-FF has the same co-location attack probability upper bound as CLR.

In Xiao et al. (2021) the designed strategy minimizes co-located VMs' rate by evaluating malicious behavior based on the reputation of users. The reputation represents the probability that a user is an attacker. The algorithms in Long and Duong (2020), Han et al. (2017), Bijon et al. (2015), Natu and Duong (2017), Ding et al. (2018), Jia et al. (2019) and Agarwal and Duong (2019) incorporate an additional restriction on the number of users by servers. This restriction is unreliable and disrupts the multi-tenancy concept, consequently, hurts resources' optimization. In Han et al. (2017), Ding et al. (2018), Jia et al. (2019) and Agarwal and Duong (2019) the number of users by servers is an algorithm's input, while in Bijon et al. (2015), Natu and Duong (2017) it is controlled through the users interventions. In fact, the two last algorithms allow users to specify which VMs may or may not be co-located. We stress that this approach is not realistic in public cloud because the adversary can use many different accounts, in addition he could be a legitimate user communicating with the target. The placement algorithm in Berrima et al. (2016) is limited to semi-online setting since the VMs have to wait until a specific number of VMs is gathered in a queue before being randomly assigned to servers. CLR and SR-FF sound straightforward because they attempt to find a compromise between security and resource optimization, without referring to elusive constraints. However they do not provide *a priori* guarantee of an upper bound on the co-location attack probability (i.e ϵ in the co-location resistance definition in Azar et al. (2014)). Instead, they can a posteriori calculate the upper bound for a given configuration. Guaranteeing such a bound is very useful in practice, it allows the cloud manager to ensure, even in hostile environments, a certain level of security for its customers. In this way, it will be possible to establish a security level agreement between the cloud and the customer, which can be considered as a sort of service level agreements (SLA) from a security perspective.

3. Problem formulation

In a realistic cloud setting, the VM placement problem is indeed an online one in which new VM requests arrive sporadically and have to be served immediately at the beginning of each time step k by the VM placement algorithm. Let N be the set of n servers. At time step k, the available resources of each server s_i are denoted by a d-dimensional vector $C_i^k = (c_1^k, \ldots, c_d^k)$. Let M_k be the sequence of the new VM requests at time step k. Let $\sigma_{k-1} = M_0 \odot \ldots \odot M_{k-1}$ be the sequence of all VM requests up to time step $k-1$, where $\odot$ is the concatenation operator of sequences. Each VM v_i has a requirement resources' capacity denoted by a d-dimensional vector $R_i = (r_1, \ldots, r_d)$. Let $\psi_{\sigma_{k-1},N}$ be the feasible assignment, at the beginning of time step k, in which each $v_i \in \sigma_{k-1}$ is mapped to a server $s_j \in N$, denoted by $v_i \mapsto s_j$, in which all resource constraints are satisfied, i.e for $\ell = 1 \ldots d$ we have $\sum_{i'} R_{i'}[\ell] \leq C_j^{k-1}[\ell]$: $v_{i'} \mapsto s_j$.

The objective at a time step k is to extend the assignment $\psi_{\sigma_{k-1},N}$ with a new assignment $\psi_{M_k,N}$ without violating the resource constraints resulting in the updated feasible assignment $\psi_{\sigma_k,N}$, with $\sigma_k = \sigma_{k-1} \odot M_k$.

Example 3.1. Consider a set $N = \{S_1, S_2, S_3\}$ of three servers. For simplicity's sake, we assume that the three servers have one resource with the same capacity of 10 units. Let us suppose up to time $t = 3$, we have received a sequence $\sigma_3 = v_1 \odot v_2 \odot v_3$ of three VMs with $R_1[1] = 5$, $R_2[1] = 4$, and $R_3[1] = 3$. We consider the assignment $\psi_{3,N} = \{v_1 \mapsto S_1, v_2 \mapsto S_2, v_3 \mapsto S_1\}$. This assignment is feasible since the resource constraints on each server S_i is verified. Now let us assume that at time $t = 4$, a new VM v_4 arrives (i.e $M_4 = v_4$) with $R_4[1] = 3$. There are many possible assignments of v_4. Let $\psi_{\sigma_4,N}$ and $\psi'_{\sigma_4,N}$ be the assignments extending $\psi_{\sigma_3,N}$ by $v_4 \mapsto S_1$, and $v_4 \mapsto S_2$, respectively. Both assignments are feasible since $C_1^4[1] = 4 \geq R_4[1]$ and $C_2^4[1] = 6 \geq R_4[1]$.

Now, at time step k, what are the characteristics of assignment $\psi_{\sigma_k,N}$ w.r.t. resource consumption and co-location attack resistance? To address the resource consumption of assignment $\psi_{\sigma_k,N}$, one could simply count the number of used servers over time (i.e. active hosts), which correlates with an overall energy consumption. As for assessing the co-location attack resistance of assignment $\psi_{\sigma_k,N}$, one needs to have a full knowledge of the nature/type of each VM. Unfortunately, we are uncertain about the type of each VM request. Each VM can be one of three types: attack, target, or neutral (i.e. neither attack nor target). To deal with such an uncertainty, one might make use of available historical data to come up with an estimation of the probability of being an attack, target or neutral VM. We denote by $Pr(a), Pr(t), Pr(n)$ the probabilities of a VM being an attack, a target and neutral, respectively. Thus, given m VMs the number of possible scenarios, that we denote by Ω, is equal to 3^m. As a matter of fact, in each one of the 3^m scenarios of the VM types, an assignment $\psi_{\sigma_k,N}$ can either exhibit or not a co-location attack. For each scenario $\omega \in \Omega$, we denote by $Pr(\omega)$ its probability, which equals to the product of the probability of each type of VM. In each ω, we label an attacker VM by A, a target VM by T and a neutral VM by N. In this way, a scenario will be denoted by a sequence of labels in $\{A, T, N\}$ where the ith label corresponds to the ith VM type in the corresponding sequence.

More specifically, from a security perspective, an assignment exhibits a co-location attack w.r.t a fraction δ and co-location attack probability threshold ϵ if the probability that at least one of the attacker VMs co-locates with at least $\delta \times t$ does not exceed ϵ. However, the definition of such an assignment must consider the uncertainty of the VM's type. This idea is formalized by the notion of a co-location resistant (CLR) assignment. Before introducing the CLR assignment, we need to define a random variable on (Ω, Pr), which outputs 1 whenever there is a co-location attack.

Definition 3.2 (*CLA Random Variable*)**.** Given an assignment $\psi = \psi_{\sigma_k,N}$, the expected number of attacker VMs $q = Pr(a) \times |\sigma_k|$, the expected number of target VMs $t = Pr(t) \times |\sigma_k|$, the expected number of neutral VMs $b = Pr(n) \times |\sigma_k|$, and $\delta \in]0, 1]$. A Co-location Attack (CLA) random variable w.r.t $\psi_{\sigma_k,N}$ and δ, is defined by $CLA_\delta^\psi : \Omega \to \{0, 1\}$ such that:

- $\forall \omega \in \Omega$, $CLA_{\delta}^{\psi}(\omega) = 1$ if at least $\delta \times t$ target VMs are co-located with at least one of the q attacker VMs, 0 otherwise.
- $\forall y \in \{0, 1\}$, $Pr(CLA_{\delta}^{\psi} = y) := \sum_{\omega \in \Omega} Pr(\omega) : CLA_{\delta}^{\psi}(w) = y$.

Definition 3.3 (*CLR Assignment*)**.** Given an assignment $\psi = \psi_{\sigma_k,N}$, the expected number of attacker VMs $q = Pr(a) \times |\sigma_k|$, the expected number of target VMs $t = Pr(t) \times |\sigma_k|$, the expected number of neutral VMs $b = Pr(n) \times |\sigma_k|$, and $\delta, \epsilon \in]0, 1]$. We say that ψ is (δ, ϵ)-Co-Location-Resistant (CLR) if:

$$\Pr(CLA_{\delta}^{\psi} = 1) \leq \epsilon$$

Let μ denote the probability of co-location attacks w.r.t δ incurred by $\psi_{\sigma_k,N}$:

$$\mu := \Pr(CLA_{\delta}^{\psi} = 1)$$

Example 3.4. Continuing with Example 3.1 and consider the following security and uncertainty parameters: $\delta = \frac{2}{3}$, $\epsilon = \frac{1}{2}$, $Pr(a) = 0.7$, $Pr(t) = 0.3$ and $Pr(n) = 0$. Let $\Omega_3 = \{AAA, AAT, AAN, \ldots, NNN\}$ be the set of $3^3 = 27$ possible scenarios of a sequence of three VMs. By applying Definition 3.2 on $\psi_{\sigma_3,N}$, we get that (we do not consider scenarios with neutral VMs since Pr(n)=0).

$$\begin{aligned} Pr(CLA_{\delta}^{\psi_{\sigma_3,N}} = 1) &= Pr(ATT) + Pr(AAT) + Pr(TTA) + Pr(TAA) \\ &= 2 \times (0.7^2 \times 0.3) + (0.7 \times 0.3^2) \\ &= 0.42 \\ &\leq \epsilon \end{aligned}$$

It follows that $\psi_{\sigma_3,N}$ is (δ, ϵ)-CLR. Similarly, to check the (δ, ϵ)-CL-resistance of the two possible assignments $\psi_{\sigma_4,N}$ and $\psi'_{\sigma_4,N}$, we need to enumerate 3^4 possible scenarios of $\Omega_4 = \{AAAA, AAAT, \ldots, NNNN\}$. We have $\psi_{\sigma_4,N}$ is not (δ, ϵ)-CLR because: $Pr(CLA_{\delta}^{\psi_{\sigma_4,N}} = 1) = 3 \times (0.7^3 \times 0.3) + 6 \times (0.7^2 \times 0.3^2) = 0.5733$. However, $\psi'_{\sigma_4,N}$ is (δ, ϵ)-CLR because $Pr(CLA_{\delta}^{\psi'_{\sigma_4,N}} = 1) = 2 \times (0.7^3 \times 0.3) + 4 \times (0.7^2 \times 0.3^2) = 0.3822$. Note that, we do not consider scenarios with three target VMs, because in this case the number of attacked VMs is 1, which is less than $\delta \times t = \frac{2}{3} \times 3 = 2$.

Now, we can formulate the online probabilistic secure VM placement problem as follows: at the beginning of each time step k, we are given:

- N: a set of n servers where each server s_i has a d-dimensional available resources vector C_i^k, a sequence $\sigma_{k-1} = M_0 \odot \ldots \odot M_{k-1}$, and $\delta, \epsilon \in]0, 1]$;
- $\psi_{\sigma_{k-1},N}$: a feasible (δ, ϵ)-CLR assignment; and
- M_k: a sequence of new VM requests where each VM v_i is associated with a d-dimensional resource requirement vector R_i.

The objective is to find an assignment $\psi_{\sigma_k,N}$ such that:

(1) **Resource constraint**: the assignment $\psi_{\sigma_k,N}$ is feasible, i.e $\forall s_j \in N$, for $\ell = 1 \ldots d$, we have $\sum_i R_i[\ell] \leq C_j^k[\ell]$: $v_i \mapsto s_j \in \psi_{\sigma_k,N}$;
(2) **Co-location-resistance constraint**: the assignment $\psi_{\sigma_k,N}$ is a (δ, ϵ)-CLR assignment;
(3) **Resource consumption**: while minimizing the number of active servers over time.

It should be stressed out that this formulation supports the online mode where the request VMs are served immediately upon their arrivals (i.e. $|M_i| = 1$ for $i = 1..k$), and the semi-online mode where the request VMs are served by batch (i.e. $|M_i| > 1$ for $i = 1..k$). Compared to the CLR definition in Azar et al. (2014), our formulation has a significant advantage. Indeed, in CLR the bound on co-location resistance holds for the final assignment of a given set of VMs. However, our formulation guarantees the co-location resistance constraint at each step of the VMs placement.

4. Confidence-based CLR assignment constraint

Consider a feasible assignment ψ of the VMs in σ_k at a time step k to the servers. For any scenario $\omega \in \Omega$ and any δ in $]0, 1]$, the type of each VM is known, then checking the value of $CLA_{\delta}^{\psi}(\omega)$ (Definition 3.2) is straightforward. As in Example 3.4, it is possible to compute μ and decide if ψ is (δ, ϵ)-CLR according to Definition 3.3. Unfortunately, since the number of scenarios is exponential in the number of VMs, such an exhaustive approach is not feasible in practice to compute μ when σ_k gets large. For instance, the size of Ω will be approximately 3.48×10^9 for the twentieth request and 5×10^{47} for the hundredth, which is utterly inaccessible. Besides, a real-world cloud receives thousands of VM requests so that we will face an overwhelming number of scenarios.

In what follows, we show how to approximate μ by confidence intervals through using a restricted sample from Ω. Then, we explore different sampling methods and conclude by providing the details of a statistical inference procedure that enforces the co-location resistance security constraint.

4.1. Estimating μ by confidence intervals

To remedy this computational challenge, our aim is to design a *practical* placement strategy based on a sample of scenarios that is able to find an assignment ψ such that, with confidence $(1-\alpha)$, ψ is *approximately* (δ, ϵ)-CLR. We now formalize such a statistical notion and define an $(\alpha, \varphi, \delta, \epsilon)$-CLR assignment as follows:

Definition 4.1 (*Approximate CLR Assignment*)**.** Given a significance level $\alpha \in [0, 1]$ and a threshold tolerance value $\varphi \in]0, 1]$, an assignment ψ is $(\alpha, \varphi, \delta, \epsilon)$-CLR iff, there exists some $\beta \in]0, \varphi[$ such that, with confidence $1 - \alpha$, ψ is $(\delta, \epsilon + \beta)$-CLR.

The significance level α and the threshold tolerance value φ are to be set by the decision-maker to allow her to: (1) specify a level above which she will have enough confidence in the level of security of the found assignment; and (2) set the upper bound value for the acceptable deviation of the security level above ϵ.

We argue, in this paper, that one possibility to overcome the computational bottleneck of an exhaustive approach is to consider a restricted sample of the scenario population $S \subseteq \Omega$ from which one can draw an inference about the overall population, specifically the probability of co-location attacks μ and hence help us find an assignment ψ that is $(\alpha, \varphi, \delta, \epsilon)$-CLR in an efficient manner.

Definition 4.2 (*Estimating* μ)**.** Let ψ be an assignment, δ be a value in $]0, 1]$ and $\mu = Pr(CLA_{\delta}^{\psi} = 1)$ be the probability of co-location attacks w.r.t δ incurred by ψ. Given a sample $S \subseteq \Omega$, the estimated value of μ w.r.t S is defined by:

$$\mu_s = Pr(CLA_{\delta}^{\psi}|_S = 1)$$

with $CLA_{\delta}^{\psi}|_S : S \to \{0, 1\}$ is the random variable CLA_{δ}^{ψ} restricted to the sample S.

Since μ_s is computed from a restricted sample, it is an estimated value of the true value μ and hence is subject to errors. A confidence interval (CI) can be used to describe how reliable the sample estimation is. A CI estimate specifies instead a range

of values within which the parameter is estimated to lie with a certain confidence level $(1-\alpha)$. A $(1-\alpha)$% CI is of the form:

$$[\mu_s - \frac{\gamma}{2}, \mu_s + \frac{\gamma}{2}]$$

where we use $\frac{\gamma}{2}$ as an error threshold to represent values for the population parameter μ for which the difference between the parameter and the sample mean estimate μ_s is not statistically significant at the α% level.

Now, given a CI $[\mu_s-\frac{\gamma}{2}, \mu_s+\frac{\gamma}{2}]$ that estimates with confidence $(1-\alpha)$% the true value of μ from a sample S, what sound statistical inference can we draw about the relationship between the true μ and ϵ given the CI $[\mu_s - \frac{\gamma}{2}, \mu_s + \frac{\gamma}{2}]$?

Given an assignment ψ, the co-location attack probability threshold ϵ, and a sample S from which we compute a $(1-\alpha)$ CI $[\mu_s - \frac{\gamma}{2}, \mu_s + \frac{\gamma}{2}]$, we distinguish between three mutual exclusive cases: (1) when with confidence $(1-\alpha)$, ψ is (δ,ϵ)-CLR; (2) when with confidence $(1-\alpha)$, ψ is not (δ,ϵ)-CLR; and (3) when we are unable to make any sound inference whether ψ is (δ,ϵ)-CLR or not.

Lemma 4.3. *Given a $(1-\alpha)$% CI $[\mu_s-\frac{\gamma}{2}, \mu_s+\frac{\gamma}{2}]$ and $\delta,\epsilon \in]0,1]$, an assignment ψ is (δ,ϵ)-CLR with confidence $(1-\alpha)$% if*

$$\mu_s + \frac{\gamma}{2} \leq \epsilon$$

Proof. Consider a $(1-\alpha)$% CI $[\mu_s - \frac{\gamma}{2}, \mu_s + \frac{\gamma}{2}]$ and $\delta,\epsilon \in]0,1]$. Suppose that we have an assignment ψ such that $\mu_s + \frac{\gamma}{2} \leq \epsilon$. In addition, we are given $\mu_s - \frac{\gamma}{2} < \mu < \mu_s + \frac{\gamma}{2}$ with a $(1-\alpha)$% CI. Thus if $\mu_s + \frac{\gamma}{2} \leq \epsilon$, we get $\mu \leq \epsilon$ with confidence $(1-\alpha)$. Based on Definition 3.3, we conclude that if $\mu_s + \frac{\gamma}{2} \leq \epsilon$, ψ is (δ,ϵ)-CLR with confidence $(1-\alpha)$. □

Lemma 4.4. *Given a $(1-\alpha)$% CI $[\mu_s-\frac{\gamma}{2}, \mu_s+\frac{\gamma}{2}]$ and $\delta,\epsilon \in]0,1]$, an assignment ψ is not (δ,ϵ)-CLR with confidence $(1-\alpha)$% if*

$$\epsilon < \mu_s - \frac{\gamma}{2}$$

Proof. Consider a $(1-\alpha)$% CI $[\mu_s - \frac{\gamma}{2}, \mu_s + \frac{\gamma}{2}]$ and $\delta,\epsilon \in]0,1]$. Suppose that we have an assignment ψ such that $\epsilon < \mu_s - \frac{\gamma}{2}$. In addition, we are given $\mu_s - \frac{\gamma}{2} < \mu < \mu_s + \frac{\gamma}{2}$ with a $(1-\alpha)$% CI. Thus if $\epsilon < \mu_s - \frac{\gamma}{2}$, we get $\mu > \epsilon$ with confidence $(1-\alpha)$. Based on Definition 3.3, we conclude that if $\epsilon < \mu_s - \frac{\gamma}{2}$, ψ is not (δ,ϵ)-CLR with confidence $(1-\alpha)$. □

Lemma 4.5. *Given a $(1-\alpha)$% CI $[\mu_s-\frac{\gamma}{2}, \mu_s+\frac{\gamma}{2}]$ and $\delta,\epsilon \in]0,1]$, we cannot decide whether ψ is (δ,ϵ)-CLR or not with confidence $(1-\alpha)$% if*

$$\mu_s - \frac{\gamma}{2} \leq \epsilon < \mu_s + \frac{\gamma}{2}$$

Proof. Consider a $(1-\alpha)$% CI $[\mu_s - \frac{\gamma}{2}, \mu_s + \frac{\gamma}{2}]$ and $\delta,\epsilon \in]0,1]$. Suppose that we have an assignment ψ such that $\mu_s - \frac{\gamma}{2} \leq \epsilon < \mu_s + \frac{\gamma}{2}$. In addition, we are given $\mu_s - \frac{\gamma}{2} < \mu < \mu_s + \frac{\gamma}{2}$ with a $(1-\alpha)$% CI. Thus if $\mu_s-\frac{\gamma}{2} \leq \epsilon < \mu_s+\frac{\gamma}{2}$, we cannot decide whether $\mu \leq \epsilon$ or not with confidence $(1-\alpha)$. Based on Definition 3.3, we conclude that if $\mu_s-\frac{\gamma}{2} < \mu < \mu_s+\frac{\gamma}{2}$, we cannot decide whether ψ is (δ,ϵ)-CLR or not with confidence $(1-\alpha)$. □

In the case where it is impossible to make an inference, one is nevertheless still able to infer with confidence $(1-\alpha)$% the co-location-resistance for any assignment ψ by adjusting ϵ value as stated in the two following lemmas.

Lemma 4.6. *Given a $(1-\alpha)$% CI $[\mu_s-\frac{\gamma}{2}, \mu_s+\frac{\gamma}{2}]$ and $\delta,\epsilon \in]0,1]$, an assignment ψ is (δ,ϵ')-CLR with confidence $(1-\alpha)$% if*

$$\mu_s - \frac{\gamma}{2} \leq \epsilon < \mu_s + \frac{\gamma}{2}$$

where $\epsilon' = \epsilon + \gamma$

Proof. Similar to Proof 4.3. □

Lemma 4.7. *Given a $(1-\alpha)$% CI $[\mu_s-\frac{\gamma}{2}, \mu_s+\frac{\gamma}{2}]$ and $\delta,\epsilon \in]0,1]$, an assignment ψ is not (δ,ϵ')-CLR with confidence $(1-\alpha)$% if*

$$\mu_s - \frac{\gamma}{2} \leq \epsilon < \mu_s + \frac{\gamma}{2}$$

where $\epsilon' = \epsilon - \gamma$

Proof. Similar to Proof 4.4. □

4.2. Sample construction

This section is dedicated to how to construct our sample scenarios of VMs' types. Given α and γ, we will present how one can determine the appropriate sample size l from which the calculated (1-α)% CI covering the true μ has a width no more than γ.

The set of the l random variables CLA_δ^ψ corresponds to a set of Bernoulli trials. Thus, their sum follows a binomial distribution $B(l,\mu)$. Due to the central limit theorem, such binomial distribution can be approximated by a Normal distribution under the assumption that our sample size l is greater than or equal to 30. Indeed, given α and γ, one can determine the appropriate sample size from which the calculated CI has the desired width. The sample size equals

$$l = (Z_{\alpha/2}/\frac{\gamma}{2})^2 \mu_s(1-\mu_s)$$

where $Z_{\alpha/2}$ is the cutoff value for the Z-statistic. Since we have not any guess for μ_s, we maximize l and get an overestimation of the sample size, which holds regardless of the value of μ_s. We consider the maximum value of $\mu_s(1-\mu_s)$ when $\mu_s = 0.5$. Hence,

$$l = 0.25 \times (Z_{\alpha/2}/\frac{\gamma}{2})^2 \quad (1)$$

Now, we present two methods, the stratified sampling (Thompson, 2012) and a random sampling method, to generate samples from the true/exhaustive VMs type scenarios set Ω. These methods take into account the probability distribution of scenarios in Ω. They are then used to estimate the true probability of co-location attacks μ and build our confidence interval.

4.2.1. Stratified sampling

In the stratified sampling method (Thompson, 2012), we are given the attacker VM sampling probability $Pr(a)$, the target VM sampling probability $Pr(t)$ and the neutral VM sampling probability $Pr(n)$. Let S be our sample, l be its size and m the number of VMs to sample. Ω includes 3^m scenarios and each covers types for the m VMs. We partition Ω into $\frac{(m+1)(m+2)}{2}$ strata with respect to the number of observed VMs of each type. All scenarios in each stratum have an equal number of observed attacker VMs, target VMs and neutral VMs. To generate sample scenarios, we need to determine the proportion of scenarios that should have i attacker VMs and j target VMs for each $i \in \{0,\ldots,m\}$ and $j \in \{0,\ldots,i\}$. Consider a stratum with $i = k_1$ and $j = k_2$, we compute its sample size l_{k_1,k_2} as follows:

$$l_{k_1,k_2} = \lceil l \binom{m}{k_1}\binom{m-k_1}{k_2} Pr(a)^{k_1} Pr(t)^{k_2} Pr(n)^{(m-k_1-k_2)} \rceil \quad (2)$$

Thus, the sample size of each stratum is simultaneously proportional to VMs type distribution and the stratum size. Then, we apply Simple Random Sampling (SRS) (Vitter, 1984) to generate these l_{k_1,k_2} scenarios with exactly k_1 attacker VMs and k_2 target

VMs. Given a scenario, we select k_1 attacker VMs from $[1 \dots m]$ using SRS without replacement. Similarly, we randomly select k_2 target VMs from the remaining set of VMs. Consequently, the rest of VMs are labeled as targets.

4.2.2. Random sampling

Our random sampling method is similar to the sample generation of the Monte Carlo simulation (Richtmyer, 1951) . It follows the same steps to generate the sample. In this method, we assume the l previously simulated scenarios. Each scenario presents a sequence of types associated to received VMs in order. At time k, we aim to simulate l scenarios for the newly received VMs in M_k. The attack, the target and the neutral VMs may unfold with probabilities $Pr(a)$, $Pr(t)$ and $Pr(n)$ respectively. The probability distribution for the type could be simulated by picking uniformly at random a value $x \in [0, 1]$. If $x \leq Pr(a)$, the VM's type sample is an attacker VM, else if $x \leq Pr(a) + Pr(t)$, it is a target VM. Otherwise, it is a neutral VM. This ensures that each type value occurs with the required probabilities. This process would then be replicated ideally l times and each type $i \in \{1, \dots, l\}$ is concatenated to the correspondent ith past scenario. The result is a set of sample scenarios that represents the population Ω.

Example 4.8. We consider a configuration in which $Pr(a) = 0.7$, $Pr(t) = 0.3$, $Pr(n) = 0$, $\alpha = 0.5$, and $\gamma = 0.3$. The number of scenarios to simulate is $l = 6$ according to Eq. (1). Consider the time step $k = 4$, i.e., we have received three VMs, and at a beginning of time step $k = 4$ a new VM arrives. We assume that the random sampling method receives $S_3 = \{$ *AAA*, *TAT*, *TAA*, *ATT*, *ATA*, *AAA*$\}$ and generates the random values 0.88, 0.02, 0.51, 0.92, 0.73, and 0.64 in order. Accordingly, the new set of simulated scenarios is $S_4 = \{$ *AAAT*, *TATA*, *TAAA*, *AATT*, *TTAT*, *AAAA* $\}$. However, using the Stratified sampling we get $l_{0,4} = 1, l_{1,3} = 1$, $l_{2,2} = 2$ and $l_{3,1} = 2$, and hence $S_4 = \{$ *TTTT*, *TTAT*, *TTAA*, *TAAT*, *TAAA*, *TAAA* $\}$.

4.3. Enforcing the approximate CLR assignment constraint

We are now in a position to propose a statistical inference procedure, depicted in Algorithm 1, that takes as input an assignment ψ, security parameters δ and ϵ, a significance level α, and a threshold tolerance φ. The output is a Boolean *Satisfied*, which is set to false if ψ is not $(\alpha, \varphi, \delta, \epsilon)$-CLR; otherwise, *Satisfied* is set to true and β to the minimum feasible error value smaller than φ.

Algorithm 1 Statistical inference procedure

Require: ψ, α, φ, δ, ϵ
Ensure: *Satisfied*, β

1: Generate a sample S that guarantees a $(1 - \alpha)$%CI with a theoretical margin of error $\frac{\varphi}{2}$. Let the built $(1 - \alpha)$%CI be $[lb, ub]$ and its width $ub - lb \leq \frac{\varphi}{2}$.
2:

case 1 if $ub < \epsilon$, then ψ is $(\alpha, \varphi, \delta, \epsilon)$-CLR for all values of $\beta \in]0, \varphi[$. Set *Satisfied* to True and β to 0.

case 2 if $ub - \varphi < \epsilon \leq ub$, then ψ is $(\alpha, \varphi, \delta, \epsilon)$-CLR for all $\beta \in]ub - \epsilon, \varphi[$. Set *Satisfied* to True and β to $ub - \epsilon$.

case 3 if $ub - \varphi \geq \epsilon$, then ψ is not $(\alpha, \varphi, \delta, \epsilon)$-CLR. Set *Satisfied* to False and β to N/A.

In Algorithm 1, we set γ to φ in order to build our confidence interval as depicted in Step 1. Then, we deal with three cases in Step 2. Case 1 is covered by Lemma 4.3. If $ub < \epsilon$, then ψ is (δ, ϵ)-CLR with confidence $(1 - \alpha)$. Thus, ψ is $(\alpha, \varphi, \delta, \epsilon)$-CLR for all values of $\beta \in]0, \varphi[$. Case 2 is covered by Lemma 4.6 then, with confidence $(1 - \alpha)$, ψ is $(\delta, \epsilon + \beta)$-CLR for $\beta \in]ub - \epsilon, \varphi[$. Thus, ψ is $(\alpha, \varphi, \delta, \epsilon)$-CLR for all $\beta \in]ub - \epsilon, \varphi[$.

In Case 3, ψ is not (δ, ϵ)-CLR according to Lemma 4.4. When $lb - \varphi < \epsilon \leq ub - \varphi$ (case 3.1), ψ may or may not be $(\delta, \epsilon + \beta)$-CLR with confidence $(1 - \alpha)$ for $\beta \in]ub - \epsilon, \varphi[$. There exists no values of $\beta \in]ub - \epsilon, \varphi[$ since $ub - \epsilon \geq \varphi$. Nevertheless, we are still unable to decide whether ψ is $(\delta, \epsilon + \beta)$-CLR or not with confidence $(1 - \alpha)$. Hence, we assume that ψ is not $(\alpha, \varphi, \delta, \epsilon)$-CLR in this sub-case in order to guarantee the demanded security level with confidence $(1 - \alpha)$. Nevertheless, if we are willing to increase the threshold tolerance error value φ to $\frac{3\varphi}{2}$ instead, we have that ψ is $(\delta, \epsilon + \beta)$-CLR for all $\beta \in]ub - \epsilon, \frac{3\varphi}{2}[$. Thus, ψ is $(\alpha, \frac{3\varphi}{2}, \delta, \epsilon)$-CLR for all values of $\beta \in]ub - \epsilon, \frac{3\varphi}{2}[$. If $lb - \varphi \geq \epsilon$ (case 3.2), then ψ is not $(\delta, \epsilon + \beta)$-CLR with confidence $(1 - \alpha)$, which is covered by Lemma 4.7. Hence, there exists no $\beta \in]0, \varphi[$ such that ψ is $(\delta, \epsilon + \beta)$-CLR with confidence $(1 - \alpha)$. Thus, ψ is not $(\alpha, \varphi, \delta, \epsilon)$-CLR.

Note that φ is indeed a threshold value specified by the decision-maker as the acceptable deviation from ϵ. It is, then, used by our inference method to specify the width of our CI. Hence, in the inference procedure, it is associated with some error ranges. It is, thus, important to understand the different meanings of φ: the decision maker's perspective and its utilization in the construction of the CI.

5. Approximate sampling-based CLR VM placement strategy

This section is devoted to describe our co-location resistant placement strategy. It is made of three procedures used to construct samples, to check our approximate CLR constraint, and to optimize resources. Next, we describe the strategy and its procedures, followed by a proof of the strategy's correctness, a time complexity analysis, and an illustrative example.

5.1. Our placement strategy

The proposed strategy, described in Algorithm 2, is parameterized by the security parameters δ and ϵ, the uncertainty parameters $Pr(a)$, $Pr(t)$ and $Pr(n)$, and the confidence level α and error threshold value φ. In addition, it considers an optimization attempts threshold parameter θ, which bounds the degree of greediness for resource optimization. At a time k, given a set of available servers N, a new VM v, a previous $(\alpha, \varphi, \delta, \epsilon)$-CLR assignment $\psi_{\sigma_{k-1},N}$, and a set S_{k-1} of past scenarios, the algorithm outputs an $(\alpha, \varphi, \delta, \epsilon)$-CLR assignment $\psi_{\sigma_k,N}$, its corresponding error tolerance β, and the new sample S_k.

We proceed as follows; First, it generates the set of sample scenarios associated to the current VMs using the function *GetSample* (line 1). Next, it sorts the servers in N by their remaining capacity in increasing order (line 2). Then, in a *repeat* loop (lines 4–8) the two functions: *OptH* (line 5) and the *ACLR* (line 7) interact iteratively in order to make an $(\alpha, \varphi, \delta, \epsilon)$-CLR assignment minimizing as much as possible resource consumption. Note that *OptH* can be considered as a generic black box, which can be implemented by various optimization heuristics. The *repeat* loop is performed until *OptH* returns an assignment accepted by the *ACLR* or the number of iterations exceeds the predefined threshold θ (line 8). In the first case, the error tolerance β associated to the accepted assignment is returned. In the second case, v is assigned to a new empty server and *ACLR* is called again to get the associated error tolerance β (line 10–11).

Algorithm 2 Approximate sampling-based CLR VM placement strategy

Parameters
δ : attack coverage threshold.
ϵ : co-location attack probability threshold.
θ : optimization attempts threshold.
α : confidence level.
φ : margin of error.
$Pr(a)$: probability of being an attacker VM.
$Pr(t)$: probability of being a target VM.
$Pr(n)$: probability of being a neutral VM.
Require:
v: new VM
ψ_{k-1}: past assignment
S_{k-1}: past scenarios sample
N: the set of servers
Ensure:
ψ: new assignment
β: tolerance value
S_k: new sample
1: $S_k \leftarrow GetSample(v, \alpha, \varphi, Pr(a), Pr(t), Pr(n), S_{k-1})$
2: $N' \leftarrow sort(N)$
3: $i \leftarrow 0$ ▷ A counter for optimization trials.
4: **repeat**
5: $\psi \leftarrow \psi_{k-1} \cup v \mapsto OptH(v, i, N')$
6: $i \leftarrow i + 1$
7: $\beta \leftarrow \text{ACLR}(\psi, S, \varphi, \delta, \epsilon)$
8: **until** $(\beta! = \text{N/A})$ or $(i = \theta)$
9: **if** $(i = \theta)$ & $(\beta = \text{N/A})$ **then**
10: $\psi \leftarrow \psi_{k-1} \cup v \mapsto EmptyServer()$
11: $\beta \leftarrow ACLR(\psi, S, \varphi, \delta, \epsilon)$
12: **return** ψ, β, S_k

5.2. Sample construction methods

Our sample construction procedure *GetSample* inputs α and φ as well as the probabilities $Pr(a)$, $Pr(t)$ and $Pr(n)$. It uses α and φ to determine the sample size l, which equals $25 \times (Z_{\alpha/2}/\frac{\varphi}{4})^2$ according to Eq. (1). In addition, it exclusively uses $Pr(a)$, $Pr(t)$ and $Pr(n)$ to generate the l sample scenarios. This is a generic function that may implement any appropriate sampling algorithm. In the case of the random sampling, described in Algorithm 3, we only generate l sample scenarios for the new VM v (line 2–9) and then we extend the previously generated sample (line 10) to cover all VMs. Conversely, the stratified sampling regenerates the sample for all received VMs upon step time k. As depicted in Algorithm 4, for each possible number of attacker VMs i with $0 \leq i \leq k$ we loop over the possible number of target VMs j with $0 \leq j \leq i$ (line 2–3). i and j define a stratum, which includes $l_{i,j}$ scenarios (line 4). For each stratum we generate $l_{i,j}$ scenarios by randomly picking distinct positions of attacker VMs (*setA*) and target VMs (*setT*) (line 5–7). Then, we assign the correspondent sample type to each VM in the given scenario (line 8–15). Clearly, our strategy is not anticipatory, since it manages sample scenarios for VMs received in the past and the present.

5.3. ACLR: Approximate-CLR checking function

Algorithm 5 implements the procedure checking $(\alpha, \varphi, \delta, \epsilon)$-CLR constraint according to Definition 4.1 and Step 2 of our inference procedure (Algorithm 1) in Section 4. It is given a candidate assignment ψ, a set S of l scenarios over VMs received so far, the margin of error φ, the attack coverage threshold δ and the co-location attack probability threshold ϵ, returns the real tolerance value β if ψ is $(\alpha, \varphi, \delta, \epsilon)$-CLR, and N/A otherwise. For each scenario ω in S, it determines the number of targets

Algorithm 3 Random sampling algorithm

Require:
v: new VM
α : confidence level.
φ : margin of error.
$Pr(a)$: probability of being an attacker VM.
$Pr(t)$: probability of being a target VM.
$Pr(n)$: probability of being a neutral VM.
S_{k-1}: l past scenarios sample
Ensure:
S_k: new sample
1: $l \leftarrow 25 \times (Z_{\alpha/2}/\frac{\varphi}{4})^2$
2: **for** i in $1..l$ **do**
3: $r \leftarrow \text{Random}([0, 1[)$
4: **if** $r < Pr(a)$ **then**
5: $v.type \leftarrow$ attacker
6: **else if** $r < Pr(t)$ **then**
7: $v.type \leftarrow$ target
8: **else**
9: $v.type \leftarrow$ neutral
10: $S_k[i] \leftarrow S_{k-1}[i] \cup v.type$
11: **return** S_k

Algorithm 4 Stratified sampling algorithm

Require:
v: new VM
α : confidence level.
φ : margin of error.
$Pr(a)$: probability of being an attacker VM.
$Pr(t)$: probability of being a target VM.
$Pr(n)$: probability of being a neutral VM.
Ensure:
S_k: new sample
1: $l \leftarrow 25 \times (Z_{\alpha/2}/\frac{\varphi}{4})^2$
2: $c \leftarrow 1$
3: $S_k \leftarrow \emptyset$
4: **for** i in $0..k$ **do**
5: **for** j in $0..i$ **do**
6: $l_{i,j} \leftarrow \lceil l\binom{k}{i}\binom{k-i}{j} Pr(a)^i Pr(t)^j Pr(n)^{(k-i-j)} \rceil$
7: **for** ω in $1..l_{i,j}$ **do**
8: $setA \leftarrow$ generate random i positions of attackers from $[1..k]$
9: $setT \leftarrow$ generate random j positions of targets from $[1..k] \setminus setA$
10: $S' \leftarrow \emptyset$
11: **for** r in $1..k$ **do**
12: **if** $r \in setA$ **then**
13: $v.type \leftarrow$ attacker
14: **else if** $r \in setT$ **then**
15: $v.type \leftarrow$ target
16: **else**
17: $v.type \leftarrow$ neutral
18: $S' \leftarrow S' \cup v.type$
19: $S_k[c] \leftarrow S'$
20: $c \leftarrow c + 1$
21: **return** S_k

p' via the function *countTargets*(ω) (line 2–3). The function *co − located*(ω, ψ) (line 4) returns the number of co-located target VMs according to ω. Next, we compute μ_s with respect to δ (line 4–5). We feed l, μ_s and $1-\alpha$ to the function *getNormApproxCI* in order to construct the appropriate confidence interval *CI* using the normal approximation (line 6). Then, the algorithm encounters three cases. It returns $\beta = 0$ if the obtained confidence interval's upper-bound *CI.ub* is less than or equal to ϵ (line 7–8) and $\beta = CI.ub - \epsilon$ if $CI.ub - \varphi < \epsilon$ (9–10). Otherwise, it fails and returns N/A.

Algorithm 5 ACLR function

Require:
ψ: assignment
S: sample scenarios.
φ : margin of error.
δ : attack coverage threshold.
ϵ : co-location attack probability threshold.
Ensure: β: tolerance value in ϵ
1: $\mu_s \leftarrow 0$
2: **for** ω in S **do**
3: $p' \leftarrow countTargets(\omega)$
4: **if** co-located(ω, ψ)/$p' \geq \delta$ **then**
5: $\mu_s \leftarrow \mu_s + \frac{1}{|S|}$
6: $CI \leftarrow getNormApproxCI(l, \mu_s, 1-\alpha)$
7: **if** $CI.ub \leq \epsilon$ **then**
8: $\beta \leftarrow 0$
9: **else if** $CI.ub - \varphi < \epsilon$ **then**
10: $\beta \leftarrow CI.ub - \epsilon$
11: **else**
12: $\beta \leftarrow N/A$
13: **return** β

5.4. OptH: Optimization function

The function *OptH*(v, i, N), depicted in Algorithm 6, minimizes the number of used servers as much as possible whilst it respects our capacity constraint. It is an extension of the Best Fit heuristic (BF). We have selected BF because of its interesting competitive ratio[1] (Johnson et al., 1974), which proves its efficiency in stacking items in practice and hence minimizing the number of active hosts. BF heuristic selects the server having the smallest amount of remaining resource in which a given VM fits. Instead, our heuristic *OptH*(v, i, N) chooses the $(i+1)^{th}$ best server, i.e the server with $(i+1)^{th}$ smallest amount of remaining resource in which the VM v fits. Note that, *OptH*($v, 0, N$) coincides with BF heuristic since it returns the first $(0+1)^{st}$ server with the least remaining space. From our placement strategy perspective, this function simply returns a guess that is the best fit host excluding the hosts that already failed to be $(\alpha, \varphi, \delta, \epsilon)$-CLR for the current VM. After the θ attempts, our strategy returns the best fit $(\alpha, \varphi, \delta, \epsilon)$-CLR assignment if it exists.

Since the servers in N are sorted in an increasing order by their remaining amount of resource, *OptH* searches the position j of the first server that can host v. Then, the $(i+1)^{th}$ best server is determined by incrementing j by i (line 4). If j exceeds the size of the specified pool of servers, this function returns *NULL* (line 7–8).

[1] It is a commonly used measure to evaluate an online algorithm for an optimization problem. It is the worst-case ratio between the cost of the solution found by the algorithm and the cost of an optimal solution (offline) (Dósa and Sgall, 2014).

Algorithm 6 *OptH* function

Require:
v: VM
i : an integer
N: Sorted set of n servers
Ensure: $srv : i^{th}$ chosen server
1: $j \leftarrow 1$
2: **while** $j \leq n$ & $size(v) > capacity(N[j])$ **do**
3: $j \leftarrow j+1$
4: $j \leftarrow j+i$
5: **if** $j \leq n$ **then**
6: $srv \leftarrow N[j]$
7: **else**
8: $srv \leftarrow NULL$
9: **return** srv

5.5. Theoretical properties

We now state the main result proving that for any input sequence of VMs, our algorithm produces an $(\alpha, \varphi, \delta, \epsilon)$-CLR assignment.

Theorem 5.1. *Let N be a set of servers. For any sequence of VMs σ, our algorithm provides a $(\alpha, \varphi, \delta, \epsilon)$-CLR assignment $\psi_{\sigma,N}$.*

Proof. We prove by strong induction that $\forall k \in \mathbb{N}^*$, our algorithm provides an $(\alpha, \varphi, \delta, \epsilon)$-CLR assignment $\psi_{\sigma_k,N}$. Recall that $\psi_{\sigma_k,N}$ is a feasible assignment at the end of time step k and the beginning of time step $k+1$ in which for any $i \in 1 \ldots k$, $v_i \mapsto s_j$ with $s_j \in N$.

Base case $i = 1$: If $i = 1$, we assume that all servers are initially empty so that $\forall s_j \in N$, we do not have co-location attacks after the assignment $\psi_{\sigma_1,N} = v_1 \mapsto s_j$ and $\mu_s = 0$. In our algorithm, the *ACLR* function approves in one step any $s_j \in N$ given to the *ACLR* function because $ub \leq 0 + \frac{\varphi}{4}$ giving $ub < \epsilon + \varphi, \forall \epsilon, \varphi \in]0, 1]$. This implies that at time $k = 1$ our algorithm provides an $(\alpha, \varphi, \delta, \epsilon)$-CLR assignment $\psi_{\sigma_1,N} = v_1 \mapsto s_j, \forall s_j \in N$. So, the theorem holds when $k = 1$.

Inductive hypothesis: Suppose the theorem holds for all values of i up to some k, i.e., $\psi_{\sigma_k,N}$ is an $(\alpha, \varphi, \delta, \epsilon)$-CLR assignment, $k \geq 1$.

Inductive step: Let $i = k+1$. $\psi_{\sigma_{k+1},N} = \psi_{\sigma_k,N} \odot v_1 \mapsto s_j$ such that $\exists s_j$ that, according to our algorithm, can be (1) returned by the *optH* function at one of the θ attempts of optimization and then the $\psi_{\sigma_{k+1},N}$ is approved by the *ACLR* function. Thus, $\psi_{\sigma_{k+1},N}$ is an $(\alpha, \varphi, \delta, \epsilon)$-CLR assignment. Otherwise, when one cannot provide an assignment that is approved by the *ACLR* function for the θ attempts, (2) a new empty server is assigned to v_{k+1} so that there is no possible extra co-location attacks added to the current assignment. Thus, $\psi_{\sigma_k,N}$ is still an $(\alpha, \varphi, \delta, \epsilon)$-CLR assignment. Hence, $\psi_{\sigma_{k+1},N}$ is an $(\alpha, \varphi, \delta, \epsilon)$-CLR assignment. By construction, our algorithm provides an $(\alpha, \varphi, \delta, \epsilon)$-CLR assignment $\psi_{\sigma_{k+1},N}$ in both cases. So, the theorem holds for $i = k+1$. By the principle of mathematical induction, the theorem holds for all $k \in \mathbb{N}^*$. □

The time complexity of our secure placement strategy (Algorithm 2) in terms of the number of servers n, the number of VMs m, the size of sample l, and the optimization attempts threshold θ. The *GetSample* function, in the case of stratified sampling (Algorithm 4), has a complexity of $\mathcal{O}(l.m^3)$. However, it has a complexity of $\mathcal{O}(l)$ in the case of the random sampling (Algorithm 3). The *ACLR* function (Algorithm 5) has also a time complexity of $\mathcal{O}(l)$. The sort procedure (line 3) relies on the used algorithm to sort n servers. We consider Merge Sort algorithm as having a linearithmic complexity of $\mathcal{O}(n.log(n))$, moreover it

is highly parallelizable. The *OptH* function (Algorithm 6) requires $\mathcal{O}(n)$ operations. This function and *ACLR* are performed at most θ times in the repeat loop of Algorithm 2 (lines 3-6), leading to a complexity of $\mathcal{O}(\theta.(n+l))$. Therefore, the worst case asymptotic time of our secure placement strategy is $\mathcal{O}(l+n.log(n)+\theta.(n+l))$ operations using the random sampling and $\mathcal{O}(l.m^3 + n.log(n) + \theta.(n+l))$ using the stratified sampling.

5.6. Illustrative example

Returning to our running example (Example 3.1) with the security parameters of Example 3.4: $\delta = \frac{2}{3}$, $\epsilon = \frac{1}{2}$, $Pr(a) = 0.7$, $Pr(t) = 0.3$ and $Pr(n) = 0$. We setup the following parameters: $\theta = 3$, $\alpha = 0.5$ and $\varphi = 0.6$. Given the $(\alpha, \varphi, \delta, \epsilon)$-CLR assignment $\psi_{\sigma_3,N} = \{v_1 \mapsto S_1, v_2 \mapsto S_2, v_3 \mapsto S_1\}$, our VM placement strategy finds the $(\alpha, \varphi, \delta, \epsilon)$-CLR assignment $\psi'_{\sigma_4,N} = \psi_{\sigma_3,N} \cup \{v_4 \mapsto S_2\}$. To do so, it first calls *GetSample*, which generates a set of 6 scenarios $\{AAAT, TATA, TAAA, AATT, TTAT, AAAA\}$ using the random sampling. Thereafter, it calls the function *OptH* with $i = 0$. This function meets the BF strategy so that, it proposes to assign v_4 to S_1 being the first server with the least enough amount of available resources. Next, the *ACLR* verifies this assignment upon the six scenarios.

In the first scenario *AAAT*, the assignment $v_4 \mapsto S_1$ causes a complete attack such that the attack coverage equals 1, which is above δ, thus this scenario is violated. Similarly, the four following scenarios *TATA*, *TAAA*, *AATT* and *TTAT* are violated since their attack coverage are 1, 1, 1, and $\frac{2}{3}$, respectively. Conversely, the last scenario AAAA has an attack coverage 0 less than $\delta = \frac{2}{3}$ and hence it is satisfying. Thus, there are four out of six violated scenarios inducing an estimated co-location probability $\mu_s = \frac{5}{6}$ and a 50% CI [0.7307,0.936]. Since the CI's upper-bound 0.936 is greater than $\epsilon + \varphi = 0.9$, the first attempt of assigning v_4 to S_1 fails.

In the second attempt, *OptH* is invoked with $i = 1$ returning S_2. For this assignment, only the scenarios *AAAT*, *TAAA*, and *AATT* have an attack coverage that exceeds $\frac{1}{2}$. Thus the estimate co-location probability incurred by this assignment is $\mu_s = \frac{1}{2}$ and the 50% CI is [0.3623,0.6377]. Therefore, we have the CI's upper-bound 0.6377 as less than $\epsilon+\varphi = 0.9$. As a result, the assignment of v_4 to S_2 is approved. As a side note, we suppose that this assignment is also rejected. Note that $i = 2$ has not yet reached θ. Then, we will try the following server according to *OptH*. If that server also fails, we have no more attempts to recall *OptH* because $k = \theta$. As a result, v_4 will be systematically put on a new free server.

6. Validation of the approach

The validation for such a statistical-based approach is a crucial aspect. Therefore, we validate, in this section, the main component of our approach that is the security constraint procedure (*ACLR*). We aim to address two questions: (i) Is our procedure statistically sound, i.e, performs at least the expected $(1-\alpha)$% right decisions (satisfying/ not satisfying) taking into account a given assignment? (ii) Have we correctly delimited the expected error of our security procedure in Theorem 6.1? And how does it look like?

We consider a cloud configuration with four servers each one with a resource capacity 1k, and twenty VMs such that $Pr(a) = \frac{1}{20}$, $Pr(t) = \frac{2}{20}$ and $Pr(r) = \frac{17}{20}$. We assume that $\delta = 0.5$. Let ψ be the assignment of these VMs such that each five VMs co-locate at the same server. We exhaustively generate Ω, which contains the 3^{20} possible scenarios of VMs types. The length of each scenario is twenty. Recall that the probability of each scenario is the product of the probabilities of its observed types. For each scenario, we compute the number of targets co-locating at least one attacker VM with respect to ψ, divided by the total number of targets in this scenario. Let the resulting value be denoted by v. μ is the sum of the scenarios' probabilities where $v \geq \delta$. In the considered configuration we obtained the true $\mu = 0.1941$. Then, we vary ϵ and φ using some step increments to generate a variety of configurations. Table 1 shows the appropriate sample size according to $< \epsilon, \varphi >$. For each configuration $< \epsilon, \varphi >$, we run our security constraint procedure 1000 times. Each time we evaluate its decision for the assignment classification against the truth as follows: (i) If $\mu < \epsilon + \varphi$ then we are right if we return "satisfying" and $\mu \leq \epsilon + \beta$; Else we are wrong. (ii) If $\mu \geq \epsilon + \varphi$ then we are right if we return "not satisfying"; Else we are wrong.

Table 1
Sample size as a function of φ (we use $\frac{\varphi}{2}$ to generate sample and build CI).

φ	0.002	0.003	0.004	0.005	0.01
Sample size (m)	3841459	1707316	960365	614634	153659

Tables 2 and 3 expose our security constraint procedure accuracy expressed as the percentage of right decisions and β_{max}, w.r.t stratified and random sampling methods, respectively. β_{max} is the maximum value of the returned β by our procedure for a specified configuration. Here, we fix α to 0.05 so that we should have an accuracy of at least 95%. Our empirical results meet our theoretical ones. When $\epsilon \in \{\mu - \frac{3\varphi}{2}, \mu - \varphi\}$, the configurations meet for more than 95% of times the true "not satisfying" decision. When $\epsilon \in \{\mu - \frac{\varphi}{2}, \mu, \mu + \frac{\varphi}{4}, \mu + \frac{\varphi}{2}, \mu + \varphi, \mu + \frac{3\varphi}{2}\}$, the configurations meet for more than 95% of times the true "satisfying" decision. In all configurations, we perform at least 95% of right decisions except configurations with $\epsilon = \mu - \frac{3\varphi}{4}$ we make wrong decisions in approximately 30% of times, as expected. The given assignment in this case is truly "satisfying" but we potentially reject it for an unwanted extra 25% of times. This practical issue comes from the critical case 3.1 of our procedure where we are unable to decide whether an assignment is $(\alpha, \varphi, \delta, \epsilon)$-CLR or not. In order to hedge it, we chose in such a case to reject an assignment when we hesitate to judge it with $(1-\alpha)$ confidence level. Thus, the probability of the false positive error[2] increases and surpasses α. The error is unfortunately inevitable. Nevertheless, we can delimit it as stated by Theorem 6.1. Such an error is tolerable in practice since we are still sure that our procedure is satisfied iff the given assignment is $(\alpha, \varphi, \delta, \epsilon)$-CLR and hence the expected ultimate goal to provide an $(\alpha, \varphi, \delta, \epsilon)$-CLR assignment by our secure placement strategy is guaranteed.

Theorem 6.1. *Given a significance level $\alpha \in [0, 1]$, a threshold tolerance value $\varphi \in]0, 1]$, security parameters $\delta, \epsilon \in]0, 1]$ and an assignment ψ with true proportion μ, our procedure cannot guarantee an $(1-\alpha)$% inference when $\mu - \varphi < \epsilon \leq \mu - \frac{\varphi}{2}$.*

Proof. Assume we have a significance level $\alpha \in [0, 1]$, a threshold tolerance value $\varphi \in]0, 1]$, security parameters $\delta, \epsilon \in]0, 1]$ and an assignment ψ with true proportion μ. Suppose that $\mu - \varphi < \epsilon \leq \mu - \frac{\varphi}{2}$. Thus, we have $\mu < \epsilon + \varphi \leq \mu + \frac{\varphi}{2}$ and hence the correct classification for ψ should be "satisfying". Let $w = ub - lb$ be the width of the built CI.

After several experiments, $(1-\alpha)$% of confidence intervals should cover the true proportion μ, i.e, $lb \leq \mu \leq ub$. The values of lb and ub vary with respect to the generated samples. Based on the yield intervals, we sometimes estimate a "satisfying" classification, which is the "correct" decision and other times

[2] False positive error or type I error occurs when a hypothesis is rejected while it is true.

Table 2
Validation of our security constraint $(\alpha, \varphi, \delta, \epsilon)$-CLR assignment: the accuracy and β_{max} under the stratified sampling for 1000 repetitions: $\alpha = 0.05$ and $\delta = 0.5$.

parameters		ϵ								
		$\mu - \frac{3\varphi}{2}$	$\mu - \varphi$	$\mu - \frac{3\varphi}{4}$	$\mu - \frac{\varphi}{2}$	μ	$\mu + \frac{\varphi}{4}$	$\mu + \frac{\varphi}{2}$	$\mu + \varphi$	$\mu + \frac{3\varphi}{2}$
$\varphi = 0.002$	accuracy	100%	98.7%	70.8%	98.9%	100%	100%	100%	100%	100%
	β_{max}	N/A	0.002	0.002	0.002	0.001	0	0	0	0
$\varphi = 0.003$	accuracy	100%	99.3%	71.7%	99%	100%	100%	100%	100%	100%
	β_{max}	N/A	0.003	0.003	0.003	0.002	0.001	0	0	0
$\varphi = 0.004$	accuracy	100%	98.8%	70.7%	99.1%	100%	100%	100%	100%	100%
	β_{max}	N/A	0.004	0.004	0.004	0.002	0.001	0	0	0
$\varphi = 0.005$	accuracy	100%	98.9%	70.5%	99%	100%	100%	100%	100%	100%
	β_{max}	N/A	0.005	0.005	0.005	0.002	0.001	0	0	0
$\varphi = 0.01$	accuracy	100%	98.7%	74.8%	99%	100%	100%	100%	100%	100%
	β_{max}	N/A	0.01	0.01	0.01	0.004	0.003	0	0	0

Table 3
Validation of our security constraint $(\alpha, \varphi, \delta, \epsilon)$-CLR assignment: the accuracy and β_{max} under the random sampling for 1000 repetitions: $\alpha = 0.05$ and $\delta = 0.5$.

parameters		ϵ								
		$\mu - \frac{3\varphi}{2}$	$\mu - \varphi$	$\mu - \frac{3\varphi}{4}$	$\mu - \frac{\varphi}{2}$	μ	$\mu + \frac{\varphi}{4}$	$\mu + \frac{\varphi}{2}$	$\mu + \varphi$	$\mu + \frac{3\varphi}{2}$
$\varphi = 0.002$	accuracy	100%	97.8%	64.3%	97.4%	100%	100%	100%	100%	100%
	β_{max}	N/A	0.002	0.002	0.002	0.001	0	0	0	0
$\varphi = 0.003$	accuracy	100%	96.9%	69%	97.5%	100%	100%	100%	100%	100%
	β_{max}	N/A	0.003	0.003	0.003	0.002	0.001	0	0	0
$\varphi = 0.004$	accuracy	100%	97.9%	66.5%	97.4%	100%	100%	100%	100%	100%
	β_{max}	N/A	0.004	0.004	0.004	0.002	0.001	0	0	0
$\varphi = 0.005$	accuracy	100%	97.3%	65.61%	97.1%	100%	100%	100%	100%	100%
	β_{max}	N/A	0.005	0.005	0.005	0.003	0.001	0	0	0
$\varphi = 0.01$	accuracy	100%	97.6%	68.8%	97.3%	100%	100%	100%	100%	100%
	β_{max}	N/A	0.01	0.01	0.01	0.005	0.003	0	0	0

we estimate a "non satisfying" classification, which is a "wrong" decision. If $\mu - \varphi + w < \epsilon \leq \mu - \frac{\varphi}{2}$, we output "satisfying" as an estimated class. It is worth noting that the potential variation of the width of an interval w may falsify the decision. Otherwise namely when $\mu - \varphi < \epsilon \leq \mu - \varphi + w$, if $lb \leq \mu < ub - (\epsilon - (\mu - \varphi))$, we output "non satisfying". This gives $ub \geq \epsilon + \varphi$ and hence $\epsilon \leq ub - \varphi \leq \mu - \varphi + w$. Besides, if $ub - (\epsilon - (\mu - \varphi)) \leq \mu < ub$, we output "satisfying". This gives $ub < \epsilon + \varphi$ and hence $\epsilon > ub - \varphi$. Since $ub - \varphi \geq \mu - \varphi$, we get $\epsilon > \mu - \varphi$.

For the remaining α% of intervals ($\mu > ub$ or $\mu < lb$), if $ub < \mu$, we output "satisfying". Here, $\epsilon > \mu - \varphi$. If $ub \geq \mu + \frac{\varphi}{2}$, we output "non satisfying". Here, $\epsilon \leq \mu - \varphi + w$. If $\mu < ub < \mu + \frac{\varphi}{2}$, we output "satisfying" when $\mu - \varphi + w < \epsilon \leq \mu - \frac{\varphi}{2}$ and "non satisfying" when $\mu - \varphi < \epsilon \leq \mu - \varphi + w$.

All in all, when $\mu - \varphi < \epsilon \leq \mu - \frac{\varphi}{2}$ our procedure cannot guarantee a sound inference with $(1-\alpha)$% level of confidence. □

Furthermore, $\epsilon = \mu - \frac{\varphi}{2}$ is also a critical case but it is not clearly visualized in results since we mostly get $w < \frac{\varphi}{2}$. We see that β_{max} is N/A when our procedure returns "non satisfying" with 100%, otherwise it is perfectly in $[0, \varphi]$ as expected. The results show that under the stratified sampling we slightly make more correct inferences compared to random sampling when $\epsilon \in \{\mu - \varphi, \mu - \frac{3\varphi}{4}, \mu - \frac{\varphi}{2}\}$.

7. Evaluation of the approach

Earlier, we discussed the validation of our procedure assessing whether an assignment is $(\alpha, \varphi, \delta, \epsilon)$-CLR or not. The ultimate goal of this investigation is to provide the desired (δ, ϵ)-CLR assignment for the cloud manager. Therefore, we evaluate here the ability of our procedure to assess whether an assignment is (δ, ϵ)-CLR with confidence $(1 - \alpha)$% or not by assessing whether an assignment is $(\alpha, \varphi, \delta, \epsilon')$-CLR with $\epsilon' = \epsilon - \varphi$. This means that our security constraint procedure is run with $\epsilon' = \epsilon - \varphi$. Here, we evaluate the decision for the estimated assignment classification against the ground truth as follows: (i) If $\mu < \epsilon$ then we are right if we return "satisfying" and $\mu \leq \epsilon' + \beta$; Else we are wrong. (ii) If $\mu \geq \epsilon$ then we are right if we return "not satisfying"; Else we are wrong.

Accuracy and β_{max} results under the stratified sampling and the random sampling methods are summarized in Tables 4 and 5, respectively. When $\epsilon \in \{\mu - \frac{3\varphi}{2}, \mu - \varphi, \mu - \frac{3\varphi}{4}, \mu - \frac{\varphi}{2}, \mu\}$, the configurations meet for more than 95% of times the true "not satisfying" decision. Clearly, it is harder for an assignment to satisfy the (δ, ϵ)-CLR constraint than to satisfy the $(\alpha, \varphi, \delta, \epsilon)$-CLR constraint. To avoid any confusion, configurations with $\epsilon = \mu$ are estimated as "not satisfying". The accuracy value is 100% because we are comparing our procedure's estimated classification to the true assessment as a $(\alpha, \varphi, \delta, \epsilon')$-CLR assignment. With regard to $(\alpha, \varphi, \delta, \epsilon')$-CLR assignment assessment, configurations with $\epsilon = \mu$ should be truly "not satisfying" rather than being truly "satisfying" as yields an (δ, ϵ)-CLR assignment assessment. This only affects accuracy calculation but the assignment is anyway estimated as "not satisfying" with confidence at least 95%. When $\epsilon \in \{\mu + \frac{\varphi}{2}, \mu + \varphi, \mu + \frac{3\varphi}{2}\}$, the configurations meet for more than 95% of times the true "satisfying" decision. Besides, we see that when $\epsilon = \mu + \frac{\varphi}{4}$ we do not reach 95% of right decisions. This value falls into the critical interval. Following Theorem 6.1, we meet the critical case when $\epsilon' \in]\mu - \varphi, \mu - \frac{\varphi}{2}]$, which is equivalent to $\epsilon \in]\mu, \mu + \frac{\varphi}{2}]$. The comparison of both sampling methods reveals that the stratified sampling outperforms the random sampling with respect to configurations with $\epsilon \in \{\mu, \mu + \frac{\varphi}{4}, \mu + \frac{\varphi}{2}\}$.

The main finding of this approach is the security guarantee it provides based on its security constraint procedure. We succeeded in customizing it so that the overall placement strategy

Table 4
Evaluation of our security constraint $(\alpha, \varphi, \delta, \epsilon')$-CLR assignment: the accuracy and β_{max} under the stratified sampling for 1000 repetitions: $\alpha = 0.05$ and $\delta = 0.5$.

parameters		ϵ								
		$\mu - \frac{3\varphi}{2}$	$\mu - \varphi$	$\mu - \frac{3\varphi}{4}$	$\mu - \frac{\varphi}{2}$	μ	$\mu + \frac{\varphi}{4}$	$\mu + \frac{\varphi}{2}$	$\mu + \varphi$	$\mu + \frac{3\varphi}{2}$
$\varphi = 0.002$	accuracy	100%	100%	100%	100%	99.3%	71.3%	99.3%	100%	100%
	β_{max}	N/A	N/A	N/A	N/A	0.002	0.002	0.002	0.001	0
$\varphi = 0.003$	accuracy	100%	100%	100%	100%	99.1%	73.4%	98.7%	100%	100%
	β_{max}	N/A	N/A	N/A	N/A	0.003	0.003	0.003	0.001	0
$\varphi = 0.004$	accuracy	100%	100%	100%	100%	99.1%	72%	99.1%	100%	100%
	β_{max}	N/A	N/A	N/A	N/A	0.004	0.004	0.004	0.002	0
$\varphi = 0.005$	accuracy	100%	100%	100%	100%	98.8%	68.4%	99%	100%	100%
	β_{max}	N/A	N/A	N/A	N/A	0.005	0.005	0.005	0.002	0
$\varphi = 0.01$	accuracy	100%	100%	100%	100%	99.1%	72%	98.6%	100%	100%
	β_{max}	N/A	N/A	N/A	N/A	0.01	0.01	0.01	0.005	0

Table 5
Evaluation of our security constraint $(\alpha, \varphi, \delta, \epsilon')$-CLR assignment: the accuracy and β_{max} under the random sampling for 1000 repetitions: $\alpha = 0.05$ and $\delta = 0.5$.

parameters		ϵ								
		$\mu - \frac{3\varphi}{2}$	$\mu - \varphi$	$\mu - \frac{3\varphi}{4}$	$\mu - \frac{\varphi}{2}$	μ	$\mu + \frac{\varphi}{4}$	$\mu + \frac{\varphi}{2}$	$\mu + \varphi$	$\mu + \frac{3\varphi}{2}$
$\varphi = 0.002$	accuracy	100%	100%	100%	100%	98%	67.81%	97.6%	100%	100%
	β_{max}	N/A	N/A	N/A	N/A	0.002	0.002	0.002	0.001	0
$\varphi = 0.003$	accuracy	100%	100%	100%	100%	97%	67.9%	97.4%	100%	100%
	β_{max}	N/A	N/A	N/A	N/A	0.003	0.003	0.003	0.002	0
$\varphi = 0.004$	accuracy	100%	100%	100%	100%	97.2%	66.3%	96.8%	100%	100%
	β_{max}	N/A	N/A	N/A	N/A	0.004	0.004	0.004	0.002	0
$\varphi = 0.005$	accuracy	100%	100%	100%	100%	96.7%	68.7%	97.8%	100%	100%
	β_{max}	N/A	N/A	N/A	N/A	0.005	0.005	0.005	0.003	0
$\varphi = 0.01$	accuracy	100%	100%	100%	100%	97.4%	68.7%	96.4%	100%	100%
	β_{max}	N/A	N/A	N/A	N/A	0.01	0.01	0.01	0.005	0

provides a (δ, ϵ)-CLR assignment with a confidence equal to $(1 - \alpha)\%$. Last but not least, both sampling methods achieve good results in terms of accuracy.

8. Deployment verification

Our algorithm is the first of its kind in what concerns vision and design. It is the first that achieves a priori a security level specified with a confidence level, and assumes the probability's distribution of VM types. Such assumption and the related sampling techniques allow our algorithm to guarantee the approximate CLR constraint at each time step, which suits the online nature of the problem at hand. In this section, we assess these novel features when our placement strategy is deployed.

8.1. Experimental settings

To evaluate our approach in terms of security and resource optimization, we implemented our placement strategy using the Java language. In our settings, we consider three different cloud configurations that are C1, C2 and C3, depicted in Table 6. The three configurations include $1k$ servers each one with a resource capacity $c = 1k$, and 100 VMs each comes at a time step $k \in \{1 \dots 100\}$. We associate each VM with a size of resource requirement randomly generated between 0 and 100, The VM type is defined according to the probabilities of being an attack $Pr(a)$, a target $Pr(t)$, and a neutral $Pr(n)$. The values of the three probabilities for each configuration are presented in Table 6.

We consider two metrics to evaluate and quantify the security and the resource cost:

- Attack coverage (Han et al., 2014): We use the attack coverage to evaluate the security. It represents the percentage of attacked targets among the total number of targets VMs. We say that a target VM is attacked when it is co-located with at least one attacker VM.

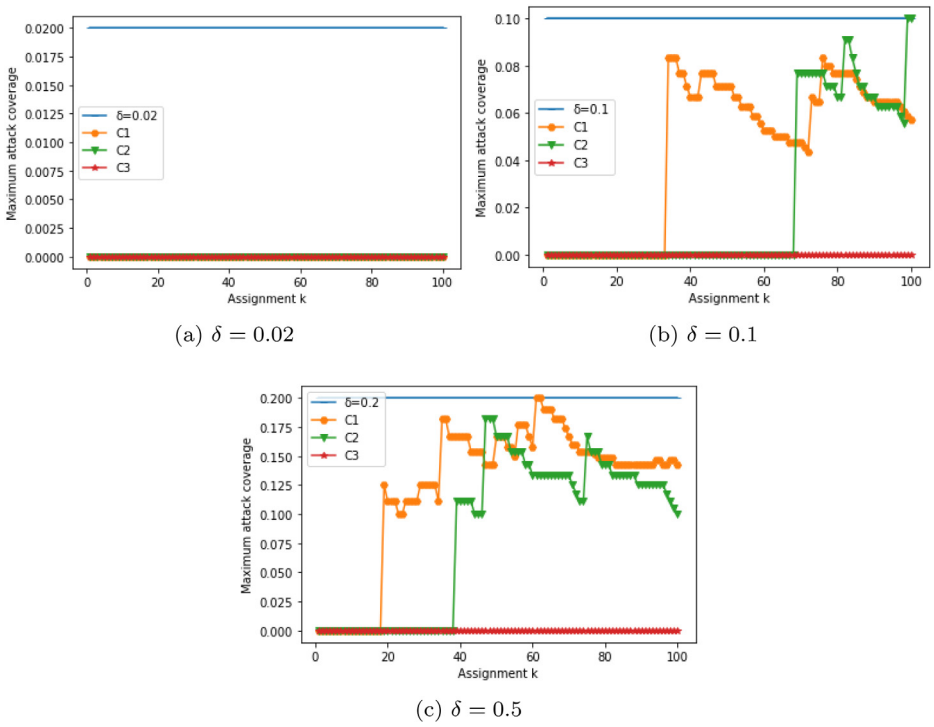
Table 6
Configurations of servers and VMs.

		C1	C2	C3
Servers	n	100	100	100
	c	1k	1k	1k
VMs	m	100	100	100
	$Pr(a)$	5%	20%	35%
	$Pr(t)$	35%	20%	5%
	$Pr(n)$	60%	60%	60%
	Size	[0,100]	[0,100]	[0,100]

- Active hosts: We use the number of used/occupied servers over time to quantify resource cost. In practice, it refers to energy consumption (Mann, 2015).

8.2. Results and discussion

At this point, we perform experimentation to highlight the potency of our strategy to guarantee a (δ, ϵ)-CLR assignment at a level of confidence $(1 - \alpha)$ throughout the placement of all VMs. In Fig. 1, we plot the maximum value of attack coverage among 50 repetitions (statistically significant) after each assignment for different configurations and values of δ. When we fix $\theta = 5$, $\epsilon = 0.01$, $\alpha = 0.05$ and $\varphi = 0.005$, we can say that the attack coverage results meet the attack coverage threshold δ. In Fig. 1(a), we have zero co-location attacks at maximum under the three configurations since δ is very small so that our strategy prohibits the attack of even one target VM. In fact, the highest number of target VMs is 35 in $C1$ and hence we are allowed to attack at most 0.7 target VMs (when all VMs are placed), which is less than one target VM. This holds for $C3$ in Fig. 1(b). The curve is stuck at 0 at the beginning for $C1$ and $C2$. Then, peaks arise in the curve when attacks occur and the curve decreases when new target VMs are



(a) $\delta = 0.02$ (b) $\delta = 0.1$

(c) $\delta = 0.5$

Fig. 1. Maximum attack coverage alongside VMs assignments using the random sampling under C1, C2 and C3.

placed. It is harder to make co-location attacks while satisfying the security constraint when $Pr(a)$ increases and $Pr(t)$ decreases. Indeed, first peaks for $C1$ arise earlier and seem less conservative in terms of co-location attacks throughout assignments. We have the same pattern in Fig. 1(c) but co-location attacks arise earlier because we have a higher value of δ. Note that in case of $C3$, $\delta = 0.2$ corresponds to the single attack and the attack coverage is stuck at 0.

Moreover, we measure the maximum value of β among 50 repetitions at each assignment such that $\delta = 0.1$ and $\epsilon \in$ {1e-4,0.001,0.01}. As shown in Fig. 2, we meet the margin of error threshold $\varphi = 0.005$ for the three configurations $C1$, $C2$ and $C3$. Going from $C1$ to $C3$, $Pr(a)$ increases whereas $Pr(t)$ decreases, and hence β decreases until it is stuck at 0 for the configuration $C3$ as shown in Fig. 2(c). This means that VMs tend to be put separately when $Pr(a)$ is large in comparison to $Pr(t)$. For the configurations $C1$ and $C2$, we have approximately the same pattern. The curve is stuck at 0 at the beginning. Then, the higher ϵ is, the earlier and faster is β variation. The peaks are due to co-location attacks whereas the curve decreases when new target VMs are observed in sample scenarios. The output values of β get bigger when ϵ gets lower.

In Fig. 3, we evaluate resource optimization by measuring the average number of active hosts (after 50 repetitions) as a function of the attack coverage threshold $\delta \in [0.02, 0.2]$. Here, we fix $\theta = 5$, $\epsilon = 1e-4$, $\alpha = 0.05$ and $\varphi = 0.005$. Results show an expected trade-off between security and resource optimization across $C1$, $C2$ and $C3$. Our placement strategy presents the same pattern under the two sampling methods as depicted in Figs. 3(a) and 3(b). When we increase the attack coverage threshold δ, the number of active servers decreases. It is worth noting that the higher $Pr(a)$ is, the worst the compromise is between security and resource optimization in the case of the random sampling. Nevertheless, our placement algorithm using the stratified sampling method is not sensitive to VMs type probability since the three curves coincide in Fig. 3(b).

Fig. 4 shows that our placement algorithm under the Stratified method gives a better compromise between security and resource optimization than the random sampling when the probability of attacker VMs increases. Indeed, our algorithm performs better when using the random method for configuration $C1$. Conversely when we consider the cloud configurations $C2$ and $C3$, the stratified sampling outperforms the random one and the gap between the correspondent curves gets larger in Fig. 4(c).

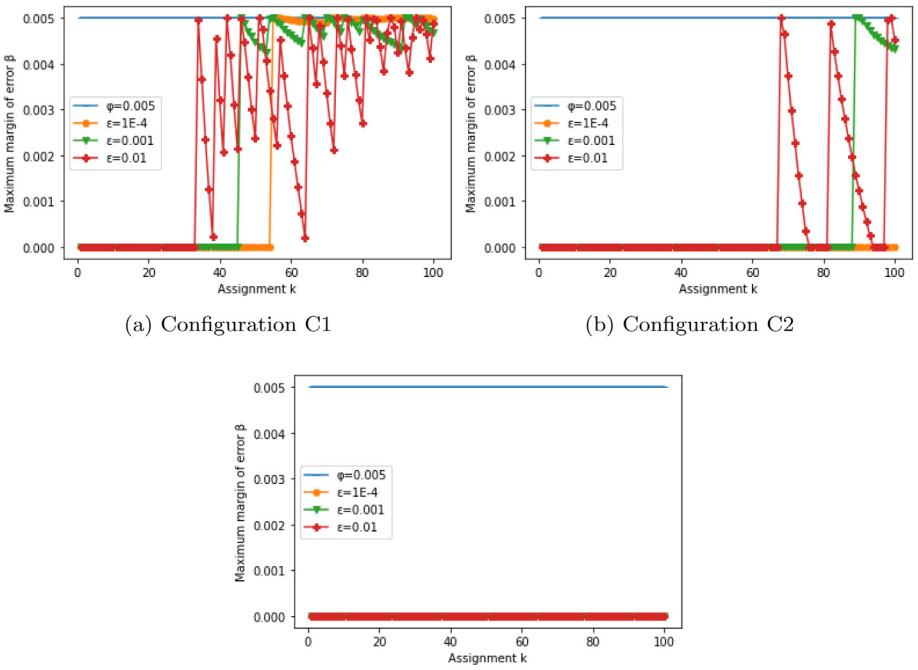
To sum-up, we showed in this section how our algorithm respects the co-location attack threshold and the margin of error specified a priori at each VM assignment. Indeed, it succeeds in providing an (α, ϵ)-CLR assignment with a level of confidence $(1 - \alpha)$ in practice. Besides, we have presented our strategy's results in terms of security and resource optimization under two sampling methods.

9. Deployment on CloudSim

By now, we have proved that our strategy guarantees the security threshold given a-priori through the validation, evaluation and deployment. Now, we deploy our placement strategy using CloudSim (Calheiros et al., 2011; Anon, 2021) on a larger scale and compare our results to the CLR method in Azar et al. (2014).

9.1. Experimental settings

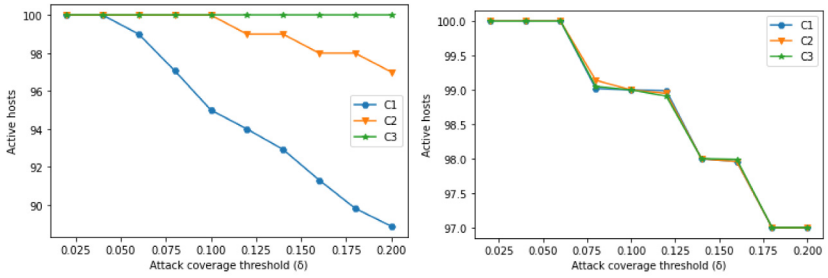
As shown in Table 7, we consider a data center with 1500 of the IBM server x3550 (2 *x* [Xeon X5675 3067 MHz, 6 cores], 16 GB) (Beloglazov and Buyya, 2012). We use the PlanetLab VMs workload (Beloglazov and Buyya, 2012) provided by default in CloudSim. These traces of VMs are collected during 10 random days in March and April 2011. They include 1516 VMs with four VM types, which are randomly given. These types determine the CPU speed, No. of CPU cores and the RAM size. Similarly, the VM types (attacker, target or neutral) are given randomly but, we set the number of targets to 20, attacker VMs to 100 and hence $Pr(t) = 0.013$ and $Pr(A) = 0.065$.



(a) Configuration C1 (b) Configuration C2

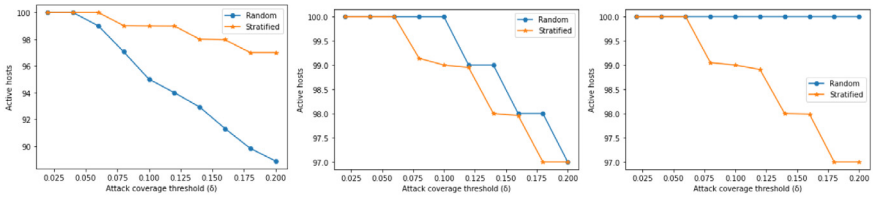
(c) Configuration C3

Fig. 2. Maximum margin of error β alongside assignments using the random sampling while varying ϵ under C1, C2 and C3.



(a) Using random sampling under C1, C2 and C3 (b) Using Stratified sampling under C1, C2 and C3

Fig. 3. Attack coverage vs. active hosts.



(a) Configuration C1 (b) Configuration C2 (c) Configuration C3

Fig. 4. Random sampling vs. stratified sampling.

Table 7
The settings of servers and virtual machines in CloudSim.

	Type	CPU speed (MIPS[a])	No. of CPU cores	RAM (MB)
Servers	1	3067	12	16384
VMs	1	2500	1	870
	2	2000	1	1740
	3	1000	1	1740
	4	500	1	613

9.2. Results

In Table 8, we assess the attack coverage and the energy consumption of our new placement strategy as a function of the attack coverage threshold $\delta \in \{0.05, 0.1, 0.25, 0.5, 0.75, 1\}$. To do this, we set $\theta = 5$, $\epsilon = 0.1$, $\alpha = 0.05$ and $\varphi = 0.005$ using the random sampling strategy. For each value of δ, the experiment is repeated 50 times. Results show the trade-off between the attack coverage and energy consumption. When we increase the attack coverage threshold δ, the consumption in terms of energy decreases. It is worth noting that we are conservative in terms of

Table 8
Our strategy: attack coverage vs. energy consumption using the random sampling policy under $\theta = 5$, $\epsilon = 0.1$, $\alpha = 0.05$ and $\varphi = 0.005$.

δ	0.05	0.1	0.25	0.5	0.75	1
Attack coverage	0.005	0.031	0.119	0.315	0.455	0.436
Energy consumption (kWh)	975.93	931.64	776.65	443.35	251.29	251.62

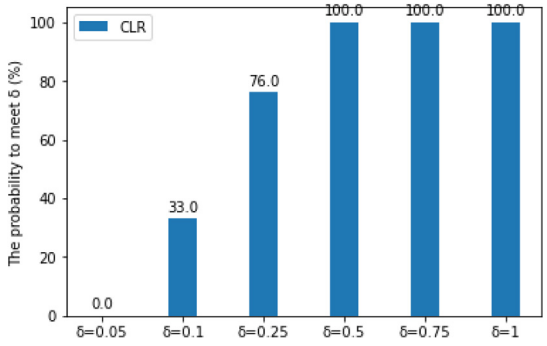


Fig. 5. The probability to meet δ among the set of possible values of λ for CLR.

attack coverage vs. the threshold δ. We can say that in practice, we guarantee the attack coverage threshold with a probability of 100%.

9.3. Comparative analysis

Our strategy has a different design compared to the existing secure strategies. Nevertheless, it is important to know where we stand empirically. Does our approach improve results alongside its novelty?

To answer this question, we select CLR (Azar et al., 2014) due to its interesting and solid formulation (Natu and Duong, 2017). Besides, it efficiently solves the co-location attack problem without elusive assumptions and shares exactly the same objectives (minimize energy consumption) as our strategy. It is pretty known that CLR cannot provide a given security level. Instead, it considers a specific parameter λ, which is the number of open servers to scatter the VMs at each time t. In order to establish a fair comparison, we implemented CLR on CloudSim and run the same experiments under various values of λ. Specifically, we tested the values of $\lambda \in \{1, 10, 20, 30, \ldots, 1490, 1500\}$ such that the step increment is 10 and $\lambda = 1500$ is the extreme case where all servers are open. For each value of $\delta \in \{0.05, 0.1, 0.25, 0.5, 0.75, 1\}$, we compute the fraction of the values of λ, which ensures a given attack coverage threshold δ. For each value of λ, the experiment is repeated 50 times. In Fig. 5, the bar chart shows the probability to meet each given δ. According to the chart, there are upward trends in the probability to meet the attack coverage threshold. We meet δ with a probability of 100% when δ is greater or equals 0.5, which is relatively a high value of attack coverage. However, CLR cannot provide a placement that meets $\delta = 0.05$ even when it meets the random policy ($\lambda = 1500$). Overall, these results demonstrate the potential superiority of our placement strategy in guaranteeing a-priori high security levels over the CLR method.

Then, we compute the average of energy consumption over the cases where λ succeed to meet the specified δ. In Fig. 6, we plot the energy consumption as a function of δ for our and the CLR placement strategies. Regarding the values of $\delta < 0.4$, CLR outperforms our strategy in terms of energy consumption optimization. Nevertheless, it meets the attack coverage threshold δ with low probabilities compared to our strategy. Otherwise, namely when $\delta \geq 0.4$, our strategy is superior to CLR. It consumes less energy while respecting the security threshold.

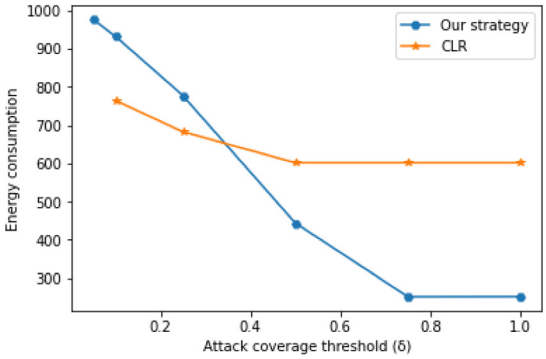


Fig. 6. Our strategy vs. CLR in terms of energy consumption.

In brief, our new approximate secure placement strategy has the advantage that it guarantees a specified level of security compared to CLR. In practice, it defeats CLR in terms of energy consumption when they both reach the same probability to ensure the attack coverage threshold.

10. Conclusion

This work focuses on the co-location attack problem arising in the IaaS layer of the public cloud. We exploited the Co-Location-Resistance (CLR) notion proposed by Azar et al. (2014) in order to introduce a novel online probabilistic secure VM placement problem. The definition of the (δ, ϵ)-CLR assignment is novel compared to previous secure placement strategies by offering the cloud provider a priori security guarantees at each VM assignment. Ensuring a priori level of security by the cloud provider for its customers is very useful in practice.

In order to solve this problem, we propose a statistical approach based on sampling and the confidence interval method to measure the reliability of the sample estimation. Thanks to sampling, we eliminated computational bottlenecks and controlled uncertain facts such as VM types. The core component of our strategy is the statistical security inference procedure, which evaluates if an assignment is a (δ, ϵ)-CLR assignment or not with a certain percentage of significance. It relies on the new statistical $(\alpha, \varphi, \delta, \epsilon)$-CLR assignment, which enables the provider to specify a confidence level α and an upper-bound deviation of security φ in addition to the security parameters δ and ϵ. The approach was validated empirically over a variety of configurations and kept errors under control. Besides, under both sampling methods we achieve good results in terms of accuracy. The proposed approach gives rise to a VM placement strategy, which is exclusively able to achieve a priori security level at a specific level of confidence. The simulation results of strategy's deployment prove their effectiveness in terms of security while optimizing resource consumption. Moreover, our approximate secure placement strategy performs more effectively than CLR in terms of energy consumption when they both respect the security constraint.

We believe that this work paves the way to a new generation of placement strategies facing the co-location attack problem. Nevertheless, more work needs to be done in the future regarding the security guarantee, the optimization profit and time delays. We intend to keep working on our strategy's security constraint in order to inform the provider and the user about the risk profile of an assignment of VMs in a given cloud and try to protect both of them from overwhelming losses in security. Besides, our strategy depends on the VMs types probabilities, which are important to the proper functioning of the security constraint. However, there is no prior work that clearly gives a solution

to provide such an information. To address this limitation, we will manage to estimate these probabilities using available historical data. For instance, we can obtain VMs types probabilities from a co-location attacks' trace based on side channel detection techniques (Narayana and Jayashree, 2020; Mushtaq et al., 2020). Another possibility is to proceed as in Xiao et al. (2021), which uses co-location attacks detecting algorithms based on behaviors to compute user reputation (Han et al., 2020; Rethishkumar and Vijayakumar, 2019).

We plan to enhance Resource optimization by serving VM requests by batch. In fact, decreasing variant of any-fit heuristic improves the results in the semi-online mode (Johnson et al., 1974). Regarding the next VMs coming at further times, we intend to sample their resource requirements. Moreover, our choice for the optimization heuristic is not definitive. Since this is a built-in function, we are going to try more heuristics, specifically stochastic ones to pick the most suitable. Another direction for future work is to involve new objectives and constraints, that map other cloud requirements such as the load balancing and VM migrations. Last but not least, there are new raising co-location attacks in the container (Shringarputale et al., 2020) and the serverless context, particularly the AWS lambdas (Yelam et al., 2021). Likewise, containers and lambdas are in a growing use in the cloud market. Therefore, our next challenge is to customize our approach to tackle such attacks while taking into consideration the specificity of containers and AWS lambdas functions.

CRediT authorship contribution statement

Marwa Thabet: Writing – original draft, Writing – review & editing, Conceptualization, Methodology, Formal analysis, Validation, Software, Investigation . **Brahim Hnich:** Writing – original draft, Writing – review & editing, Conceptualization, Methodology, Formal analysis, Validation, Supervision. **Mouhebeddine Berrima:** Writing – original draft, Writing – review & editing, Conceptualization, Methodology, Formal analysis, Validation, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Agarwal, A., Duong, T.N.B., 2019. Secure virtual machine placement in cloud data centers. Future Gener. Comput. Syst. 100, 210–222. http://dx.doi.org/10.1016/j.future.2019.05.005, URL http://www.sciencedirect.com/science/article/pii/S0167739X18326116.

Anon, 2021. CloudSim, Accessed 15 March 2021, URL http://www.cloudbus.org/cloudsim/.

Azar, Y., Kamara, S., Menache, I., Raykovau, M., Shepherdu, B., 2014. Co-location-resistant clouds. In: Proceedings Of The ACM Conference On Computer And Communications Security, Vol. 2014. pp. 9–20.

Beloglazov, A., Buyya, R., 2012. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurr. Comput. Pract. Exp. 24 (13), 1397–1420.

Berrima, M., Nasr, A.K., Ben Rajeb, N., 2016. Co-location resistant strategy with full resources optimization. In: Proceedings Of The 2016 ACM On Cloud Computing Security Workshop. In: CCSW '16, ACM, New York, NY, USA, pp. 3–10. http://dx.doi.org/10.1145/2996429.2996435, URL http://doi.acm.org/10.1145/2996429.2996435.

Bijon, K., Krishnan, R., Sandhu, R., 2015. Mitigating multi-tenancy risks in iaas cloud through constraints-driven virtual resource scheduling. In: Proceedings Of The 20th ACM Symposium On Access Control Models And Technologies. ACM, pp. 63–74.

Bobroff, N., Kochut, A., Beaty, K., 2007. Dynamic placement of virtual machines for managing SLA violations. In: 2007 10th IFIP/IEEE International Symposium On Integrated Network Management. pp. 119–128. http://dx.doi.org/10.1109/INM.2007.374776.

Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R., 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exp. 41 (1), 23–50.

Chowdhury, M.R., Mahmud, M.R., Rahman, R.M., 2015. Implementation and performance analysis of various VM placement strategies in CloudSim. J. Cloud Comput. 4 (1), 20. http://dx.doi.org/10.1186/s13677-015-0045-5.

Ding, W., Gu, C., Luo, F., Chang, Y., Rugwiro, U., Li, X., Wen, G., 2018. DFA-VMP: An efficient and secure virtual machine placement strategy under cloud environment. Peer-To-Peer Netw. Appl. 11 (2), 318–333.

Dósa, G., Sgall, J., 2014. Optimal analysis of best fit bin packing. In: International Colloquium On Automata, Languages, And Programming. Springer, pp. 429–441.

Fang, S., Kanagavelu, R., Lee, B., Foh, C.H., Aung, K.M.M., 2013. Power-efficient virtual machine placement and migration in data centers. In: 2013 IEEE International Conference On Green Computing And Communications And IEEE Internet Of Things And IEEE Cyber, Physical And Social Computing. pp. 1408–1413. http://dx.doi.org/10.1109/GreenCom-iThings-CPSCom.2013.246.

Ferreto, T.C., Netto, M.A., Calheiros, R.N., De Rose, C.A., 2011. Server consolidation with migration control for virtualized data centers. Future Gener. Comput. Syst. 27 (8), 1027–1034. http://dx.doi.org/10.1016/j.future.2011.04.016, URL http://www.sciencedirect.com/science/article/pii/S0167739X11000677.

Gupta, S., Miceli, R., Coffman, J., 2020. A replication study to explore network-based co-residency of virtual machines in the cloud. In: International Conference On Cloud Computing. Springer, pp. 1–16.

Han, Y., Chan, J., Alpcan, T., Leckie, C., 2014. Virtual machine allocation policies against co-resident attacks in cloud computing. In: IEEE Transactions On Dependable And Secure Computing. pp. 786–792.

Han, Y., Chan, J., Alpcan, T., Leckie, C., 2017. Using virtual machine allocation policies to defend against co-resident attacks in cloud computing. IEEE Trans. Dependable Secur. Comput. 14 (1), 95–108.

Han, J., Zang, W., Yu, M., Sandhu, R., 2020. Quantify co-residency risks in the cloud through deep learning. IEEE Trans. Dependable Secur. Comput..

Hasan, M.M., Rahman, M.A., 2020. A signaling game approach to mitigate co-resident attacks in an IaaS cloud environment. J. Inf. Secur. Appl. 50, 102397.

Jia, H., Liu, X., Di, X., Qi, H., Cong, L., Li, J., Yang, H., 2019. Security strategy for virtual machine allocation in cloud computing. Procedia Comput. Sci. 147, 140–144.

Johnson, D.S., Demers, A., Ullman, J.D., Garey, M.R., Graham, R.L., 1974. Worst-case performance bounds for simple one-dimensional packing algorithms. SIAM J. Comput. 3 (4), 299–325.

Levitin, G., Xing, L., 2021. Data resilience under co-residence attacks in cloud environment. In: Handbook Of Advanced Performability Engineering. Springer, pp. 739–760.

Levitin, G., Xing, L., Xiang, Y., 2020. Optimization of time constrained N-version programming service components with competing task execution and version corruption processes. Reliab. Eng. Syst. Saf. 193, 106666.

Levitin, G., Xing, L., Xiang, Y., 2021. Minimization of expected user losses considering co-resident attacks in cloud system with task replication and cancellation. Reliab. Eng. Syst. Saf. 107705.

Liang, X., Gui, X., Jian, A., Ren, D., 2017. Mitigating cloud co-resident attacks via grouping-based virtual machine placement strategy. In: Performance Computing And Communications Conference (IPCCC), 2017 IEEE 36th International. IEEE, pp. 1–8.

Long, V.D., Duong, T.N.B., 2020. Group instance: Flexible Co-location resistant virtual machine placement in IaaS clouds. In: 2020 IEEE 29th International Conference On Enabling Technologies: Infrastructure For Collaborative Enterprises. WETICE, IEEE, pp. 64–69.

Mann, Z.Á., 2015. Allocation of virtual machines in cloud data centers - a survey of problem models and optimization algorithms. ACM Comput. Surv. 48, 11:1–11:34.

Mell, P., Grance, T., 2011. The NIST Definition of Cloud Computing (Draft) Recommendations of the National Institute of Standards and Technology, Vol. 145. Nist Special Publication, pp. 1–7.

Mills, M.P., 2013. The Cloud Begins with Coal. Digital Power Group.

Mushtaq, M., Bricq, J., Bhatti, M.K., Akram, A., Lapotre, V., Gogniat, G., Benoit, P., 2020. Whisper: A tool for run-time detection of side-channel attacks. IEEE Access 8, 83871–83900.

Narayana, K., Jayashree, K., 2020. Survey on cross virtual machine side channel attack detection and properties of cloud computing as sustainable material. Mater. Today Proc..

Natu, V., Duong, T.N., 2017. Secure virtual machine placement in infrastructure cloud services. In: 2017 IEEE 10th Conference On Service-Oriented Computing And Applications. SOCA, 00, pp. 26–33. http://dx.doi.org/10.1109/SOCA.2017.12, URL.

Pires, F.L., Barán, B., 2015. Virtual machine placement literature review. CoRR, arXiv:1506.01509.

Rethishkumar, S., Vijayakumar, R., 2019. Defender Vs attacker security game model for an optimal solution to Co-resident DoS attack in cloud. In: Intelligent Communication Technologies And Virtual Mobile Networks. Springer, pp. 527–537.

Richtmyer, R.D., 1951. The Evaluation of Definite Integrals, and a Quasi-Monte-Carlo Method Based on the Properties of Algebraic Numbers. Tech. rep., Los Alamos Scientific Lab.

Ristenpart, T., Tromer, E., Shacham, H., Savage, S., 2009. Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds. In: Proceedings Of The 16th ACM Conference On Computer And Communications Security. In: CCS '09, ACM, New York, NY, USA, pp. 199–212. http://dx.doi.org/10.1145/1653662.1653687, URL http://doi.acm.org/10.1145/1653662.1653687.

Seiden, S.S., 2002. On the online bin packing problem. J. ACM 49 (5), 640–671. http://dx.doi.org/10.1145/585265.585269, URL http://doi.acm.org/10.1145/585265.585269.

Shringarputale, S., McDaniel, P., Butler, K., La Porta, T., 2020. Co-residency attacks on containers are real. In: Proceedings Of The 2020 ACM SIGSAC Conference On Cloud Computing Security Workshop. In: CCSW'20, Association for Computing Machinery, New York, NY, USA, pp. 53–66. http://dx.doi.org/10.1145/3411495.3421357.

Thompson, S., 2012. Sampling. John Wiley & Sons, Hoboken, N.J.

Varadarajan, V., Zhang, Y., Ristenpart, T., Swift, M.M., 2015. A placement vulnerability study in multi-tenant public clouds. In: USENIX Security Symposium. pp. 913–928.

Vitter, J.S., 1984. Faster methods for random sampling. Commun. ACM 27 (7), 703–718.

Wu, Z., Xu, Z., Wang, H., 2012. Whispers in the hyper-space: High-speed covert channel attacks in the cloud. In: Presented As Part Of The 21st USENIX Security Symposium. USENIX Security 12, USENIX, Bellevue, WA, pp. 159–173, URL https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/wu.

Xiao, Y., Liu, L., Ma, Z., Wang, Z., Meng, W., 2021. Defending co-resident attack using reputation-based virtual machine deployment policy in cloud computing. Trans. Emerg. Telecommun. Technol. e4271.

Xu, Z., Wang, H., Wu, Z., 2015. A measurement study on co-residence threat inside the cloud. In: USENIX Security Symposium. pp. 929–944.

Xu, Z., Wang, H., Xu, Z., Wang, X., 2014. Power attack: An increasing threat to data centers. In: NDSS.

Yelam, A., Subbareddy, S., Ganesan, K., Savage, S., Mirian, A., 2021. Coresident evil: Covert communication in the cloud with lambdas. In: Proceedings Of The Web Conference 2021. pp. 1005–1016.

Zhang, W., Jia, X., Wang, C., Zhang, S., Huang, Q., Wang, M., Liu, P., 2016. A Comprehensive Study of Co-residence Threat in Multi-tenant Public PaaS Clouds, Vol. 9977. pp. 361–375. http://dx.doi.org/10.1007/978-3-319-50011-9_28.

Zhang, Y., Juels, A., Reiter, M.K., Ristenpart, T., 2014. Cross-tenant side-channel attacks in paas clouds. In: Proceedings Of The 2014 ACM SIGSAC Conference On Computer And Communications Security. In: CCS '14, ACM, New York, NY, USA, pp. 990–1003. http://dx.doi.org/10.1145/2660267.2660356, URL http://doi.acm.org/10.1145/2660267.2660356.

Marwa Thabet is currently a Ph.D. in Computer Science at the Higher Institute of Computer Science and Communication Techniques of Hammam Sousse (ISITCom) at the University of Sousse (since 2019, Tunisia). She holds a Bachelor of Science degree in Computer Science from Higher Institute of Computer Science and Mathematics of Monastir (ISIMM) at the University of Monastir (2016, Tunisia) and was awarded for being the laureate of her promotion. Besides, she holds a Master in Automatic Reasoning Systems degree from Faculty of Sciences of Monastir (FSM) at the University of Monastir (2019, Tunisia). Her research interests include cloud computing (IaaS), co-location attack problem, optimization problems, constraint programming, artificial intelligence, data science and big data.

Brahim Hnich is currently an Associate Professor of Computer Science (CS) at the department of Computer Science at Monastir University and the director of its research Masters programme. He holds a Bachelor of Science degree in Computer Engineering from Bilkent University (1997, Turkey), a Ph.D. in Computer Science from Uppsala University (2003, Sweden), Docentship in CS from Uppsala University (2004, Sweden), as well as a HDR in Computer Science from Montepellier II University (2008, France). After his doctoral studies, He worked as Research Fellow at the Cork Constraint Computation Center (2003–2006, Ireland). Then he joined Izmir University of Economics (Izmir, Turkey) as an associate Professor (2006–2009) and was a visiting Professor in Taif University (2015–20156, KSA). He was an associate editor of the Artificial Intelligence Journal and the coauthor of more than 40 journal papers covered by Science Citation Index and more than 50 international conference papers. Two of his research papers have been selected among the best papers at the European Conference on Artificial Intelligence in 2004 and at the International Joint Conference on Artificial Intelligence in 2005. In 2004, Hnich has been nominated by University College Cork and has been shortlisted (among the 8 finalists out of 41 nominees worldwide) for the President of Ireland Young Researcher Award 2005. In 2003, Hnich's Ph.D. Thesis was nominated for the European Coordinating Committee for Artificial Intelligence's best thesis award. His research interests include, among others, artificial intelligence, (stochastic) constraint programming, global constraints, and symmetry in combinatorial problems.

Mouhebeddine Berrima received his Ph.D. in Computer Science at the University of El-Manar, Tunisia. He is an associate professor at the Department of Computer Science, University of Monastir, Tunisia. His research interests include formal verification of security protocols, security in Cloud Computing, and algebraic specifications of programs and data structures.