基于 Shamir 的虚拟机放置策略

田俊峰，王子龙，何欣枫，李珍
（河北大学网络空间安全与计算机学院，河北 保定 071002）

摘 要：为减轻云环境下同驻攻击的危害，提出了基于 Shamir 的虚拟机放置策略，并设计了与其相适应的虚拟机放置框架，通过区块链保证了所提放置策略中关键数据的安全性。该虚拟机放置策略可以有效提升虚拟机的安全性和云环境的负载均衡能力，减少资源浪费。最后通过仿真实验证明了方案的有效性。

关键词：同驻攻击；虚拟机放置策略；多目标优化；区块链
中图分类号：TP309.5
文献标识码：A
doi: 10.11959/j.issn.1000−436x.2019141

Shamir-based virtual machine placement policy

TIAN Junfeng, WANG Zilong, HE Xinfeng, LI Zhen
School of Cyber Security and Computer, Hebei University, Baoding 071002, China

Abstract: In order to alleviate the harm of co-location attack in cloud environment, a virtual machine placement strategy based on Shamir was proposed, and a virtual machine placement framework was designed. The security of key data was ensured by the blockchain in the proposed placement strategy. The virtual machine placement policy could effectively improve the security of the virtual machine and the load balancing capability of the cloud environment, and reduce resource waste. Finally, the effectiveness of the scheme is proved by simulation experiments.

Key words: co-resident attack, virtual machine placement, multi-objective optimization, block chain

1 引言

云计算平台为用户提供动态可扩展的计算和存储资源，用户间通过虚拟机共享这些资源，这种方式在提高物理资源利用率的同时，也带来了新的安全问题。共享物理资源的虚拟机之间（用户间）本该是逻辑独立的，而近来的研究表明，恶意用户可以通过建立侧信道等方式绕过逻辑独立，从而获取其他用户的私密信息，这种行为被称为“同驻攻击”。

“同驻攻击”是云环境下危害最大的攻击方式之一，恶意用户通过对特定目标发起“同驻攻击”，可获取的信息包括从粗粒度的网络流量[1]到细粒度的加密密钥[2]。文献[1]的实验表明 Cache 利用率对于 Cache 读写操作的时间有很大影响，攻击者可以通过 Prime+Probe 技术[3-4]推断受害者 Cache 使用情况及负载信息，也可以估计受害者网络流量。而网络流量这类看似没有意义的数据，或许会成为攻击者最大化攻击效果的有力武器。文献[2]指出攻击者可以利用共享硬件资源（如指令缓存）窃取加密密钥，还指出了如何处理核心迁移并确定操作是否与受害者相关联、过滤硬件和软件噪声，并以足够的频率重新获得对目标 CPU 核心的访问。

公有云上的同驻攻击根据攻击者动机及需求的不同可以分为 2 类：有目标同驻攻击与无目标同驻攻击。对于前者，攻击者有明确的目标，可能是商业竞争对手或是知名的线上服务，他们的意图往往是恶意的，如通过侧信道获取目标的私密信息

收稿日期：2019−04−24；**修回日期**：2019−07−11
基金项目：国家自然科学基金资助项目（No.61802106）
Foundation Item: The National Natural Science Foundation of China (No.61802106)

（如加密密钥），或是通过 DDoS 攻击使目标服务瘫痪。对于后者，攻击者没有特定目标，他们的动机一般是为自身牟利，比如，通过释放他人正在使用的资源来免费提升自身可用资源。有目标同驻攻击相比无目标同驻攻击而言，其攻击手段更加复杂，更难抵御，危害也更大。因此，本文关注有目标同驻攻击。

为了减轻同驻攻击的危害，本文从虚拟机放置策略的角度出发，提出了基于 Shamir 秘密共享的虚拟机放置策略（SVMPP, Shamir-based virtual machine placement policy），主要贡献如下。

1) 设计了虚拟机放置的两级管理框架，提高了虚拟机放置效率。

2) 引入 Shamir 秘密共享方案，提高了虚拟机的安全性。

3) 对虚拟机安全性、物理机的负载均衡和能源浪费进行多目标优化，在提高虚拟机安全性的同时提升了云环境的负载均衡能力，减少了资源浪费。

4) 在两级管理框架中，通过区块链技术保证了 SVMPP 实施过程的安全性。

2　相关工作

为发动同驻攻击，恶意用户需要首先尝试与目标用户同驻，然后对目标用户发动同驻攻击。在这个过程中，阻止任意一步的发生都能在一定程度上减轻同驻攻击的危害。因此，将同驻攻击抵御方案分为以下 2 类，并分别进行介绍。

1) 增加同驻难度

为了与目标同驻，攻击者通常需要启动大量虚拟机，通过同驻检测来鉴别是否与目标同驻成功，之后重复这个过程直到同驻成功。因此，为防止恶意用户与其目标同驻，可以采取以下对策。

① 增加同驻鉴别难度。最简单的鉴别同驻的方式是基于网络的方法[1]，通过使用 TCP 的 Traceroute 方法获取当前物理机的 DOM 0 IP，如果相同则证明同驻成功。云服务提供商（CSP, cloud service provider）可以通过对租户隐藏 DOM 0 IP 使攻击者需要付出更高的代价来鉴别同驻。然而新型的同驻鉴别方法层出不穷。Inci 等[5]提出一种基于 LLC（last level cache）访问冲突的隐蔽信道通信方式用于虚拟机同驻检测。Chen 等[6]设计了一种高效检测同驻的方法——Sift，通过预过滤程序显著地减少了同驻检测时间。Bates 等[7]设计了一种同驻水印的方法进行虚拟机同驻检测。这些鉴别方法快速有效，便于攻击者使用。

②周期性迁移虚拟机。Li 等[8]通过 VCG（Vickrey–Clarke–Groves）机制周期性地迁移虚拟机，使攻击者与目标同驻的概率降低，但是，因为迁移过程要占用大量的系统资源和网络带宽，所以频繁地迁移虚拟机会造成服务器性能下降，直接损害用户与云服务提供商的利益，可能导致云服务提供商违反他们的 SLA（service level agreement）协议。

③制定安全的虚拟机放置策略。Azar 等[9]最早提出利用虚拟机放置策略减轻同驻攻击危害，他们采用 CLR（co-location resistent）策略将所有的服务器分成 2 类：打开状态 N_{open} 和关闭状态 N_{close}。打开状态意味着该服务器可以接受更多的虚拟机请求，关闭状态则意味着无法接受更多的请求。CLR 维持着固定数量的 N_{open} 服务器，当虚拟机放置时，随机挑选其中一台状态为 N_{open} 的服务器进行放置，这种方法下同驻攻击的难度与 N_{open} 数量成正比，最好的状态是全部服务器都打开，这种情况类似随机放置策略，而随机放置策略无法对资源的使用进行优化。Han 等[10]提出的 PSSF（previously-selected-servers-first）虚拟机放置策略优先选择用户使用过的或者正在使用的物理机，同时限制了用户所使用的物理机数量，导致单个用户的虚拟机放置可能过于集中，从而导致恶意用户与目标同驻一台物理机上便可获得较高的收益。Afoulki 等[11]通过用户制定的安全策略保障用户的安全，该方案通过让用户指定禁止与其虚拟机同驻的其他用户名单来达到用户的安全需求，然而现实中，用户无法将所有可能的恶意用户列在名单中，故不符合实际情况。Berrima 等[12]提出使用混合队列并随机扰乱虚拟机的放置顺序，从而加大恶意用户与目标同驻的难度，然而这种方法需要维护一个虚拟机请求序列，当序列满时再进行随机扰乱，这就可能导致用户的虚拟机请求被搁置，降低了云的灵活性。

2) 增加攻击难度

①消除侧信道。由于侧信道攻击对高分辨率时钟极为依赖，故文献[3]通过移除此时钟或者增加可疑操作的等待时间[4]来防范此类攻击。然而这种方法只针对特定类型的攻击，不具有可扩展性。文献[13-16]指出虚拟机监视器的易受攻击性，一旦攻击

者攻破虚拟机监视器，则所有与其运行在相同物理机上的虚拟机都将受到威胁，文献[17]则通过移除虚拟机监视器来抵御此类攻击。

② 加强租户隔离。共享物理资源的虚拟机之间本该是逻辑独立的，但由于恶意用户通过各种方法绕过逻辑独立，故有研究提出更加强有力的租户隔离机制。石勇等[18]提出采用信息流策略控制不同虚拟机之间的信息流动，从而提高虚拟机的安全性。

3 虚拟机放置策略

3.1 Shamir 的(*k*,*n*)门限秘密共享方案

Shamir 的(*k*,*n*)门限秘密共享方案通过秘密共享算法将秘密分为 *n* 个部分，每个参与者持有其中的一部分，如果满足：1) 持有任意 *k* 个部分则可以将秘密还原；2) 持有任何少于 *k* 个部分都无法还原秘密，这种方案被称为 Shamir 的(*k*,*n*)门限秘密共享方案，*k* 为方案的门限值。

3.2 威胁模型及安全假设

恶意用户从其目标虚拟机窃取信息前，首先要与其目标同驻，为此，恶意用户要么启动尽可能多的虚拟机，要么利用虚拟机放置策略存在的漏洞。例如，在 FirstFit[19]放置策略下，恶意用户可以通过与目标虚拟机同时发送请求提高与目标的同驻概率。因此，本文假设恶意用户可以利用虚拟机放置策略存在的漏洞。为方便讨论，进行如下假设。

1) 云服务提供商拥有充足的资源。

2) 不考虑虚拟机的动态迁移，虚拟机被分配到物理机后将会一直运行，直到虚拟机关机或者被用户关闭。

3) 云服务提供商对攻击者和普通用户没有任何的先验知识，因此平等对待所有的虚拟机请求。

3.3 虚拟机放置框架

现有虚拟机放置框架一般为数据中心−服务器框架，如图 1 所示。用户将虚拟机请求发送至数据中心后，由数据中心根据虚拟机放置策略为其选择合适的物理机，这种虚拟机放置框架使管理者可以方便地管理和维护云平台，却不能很好地保障用户虚拟机的安全性。首先，如果数据中心被恶意攻击，则整个平台都将处于危险之中。其次，数据中心处理所有事务，这使数据中心的性能成为云平台效率的瓶颈，且容易造成单点失效。

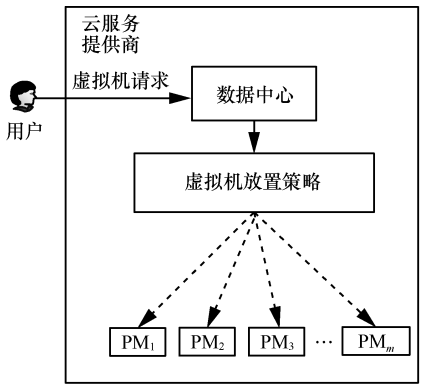


图 1 虚拟机放置框架

用户的多个秘密信息经过不同(*k*,*n*)的 Shamir 方案后会产生多组不同的信息，对应着多组虚拟机，为满足(*k*,*n*)不同而产生不同的限制要求，SVMPP 设计了两级管理框架，在如图 1 所示的虚拟机放置框架基础上，将所有服务器分组，并将用户每组虚拟机放置在不同的组内，如图 2 所示。引入组管理者负责虚拟机的放置，该过程中依据用户的 Shamir 门限值限制用户间的同驻，并根据多目标优化选择合适的物理机。

3.4 SVMPP 的多目标优化

云环境下，虚拟机放置策略一般由云服务提供商实施（如图 1 数据中心），用户并不实际参与虚拟机的放置过程。不同的是，SVMPP 的实施需要用户与 CSP 两方协作。1) 用户方面：SVMPP 要求用户将私密信息分为多份并放置在不同的虚拟机中。2) CSP 方面：CSP 要根据用户设定的 Shamir 门限值限制用户间的同驻。通过用户与 CSP 的协作可以提升虚拟机的安全性。而仅提升安全性对于用户和 CSP 而言都是不够的。尤其对于 CSP，其更关心负载均衡及资源浪费问题。SVMPP 将安全性、负载均衡和资源浪费作为目标，进行了多目标优化。SVMPP 的多目标优化问题描述如下。

假设云环境下有 *h* 个用户 $H=\{U_1,U_2,U_3,\cdots,U_h\}$，*m* 台物理机 $M=\{\mathrm{PM}_1,\mathrm{PM}_2,\mathrm{PM}_3,\cdots,\mathrm{PM}_m\}$，其中 $M_o\subseteq M$ 是开启的物理机集合，*r* 台虚拟机 $R=\{\mathrm{VM}_1,\mathrm{VM}_2,\mathrm{VM}_3,\cdots,\mathrm{VM}_r\}$，每台虚拟机 VM_i 的资源需求为 $(\mathrm{VM}_i^{\mathrm{CPU}},\mathrm{VM}_i^{\mathrm{MEM}})$。$a_{ij}$ 表示虚拟机 *i* 与物理机 *j* 之间的关系，当虚拟机 VM_i 放置在物理机 PM_j 上时 $a_{ij}=1$，否则 $a_{ij}=0$。类似地，c_{ij} 表示用户与物理机间的关系，当用户 U_i 有虚拟机放置在物理机 PM_j 上时 $c_{ij}=1$，否则 $c_{ij}=0$。为了下文叙述方便，表 1 对文中使用的主要符号进行了归纳。

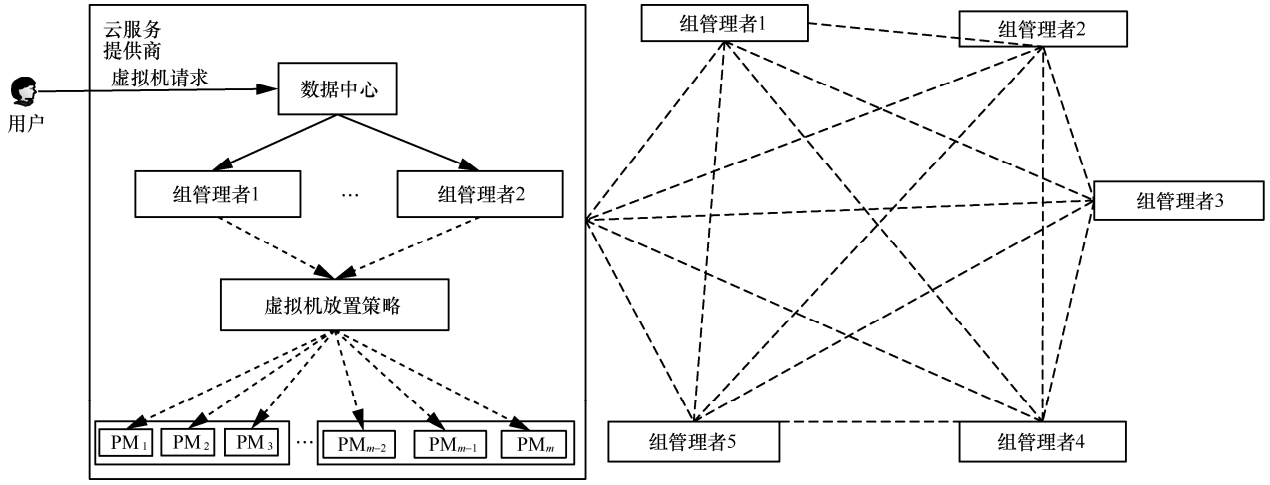


图 2　SVMPP 的两级管理框架

表 1　**本文使用的主要符号**

符号	描述
M	全部物理机的集合
M_o	开启的物理机集合
N	所有虚拟机的集合
H	所有用户的集合
U	普通用户，其启动的虚拟机集合是攻击者的目标集合
$a_{ij}\in\{0,1\}$	若 VM_i 放置在 PM_j 上，则 a_{ij}=1，否则 a_{ij}=0
$c_{ij}\in\{0,1\}$	若用户 U_i 有虚拟机放置在 PM_j 上，则 c_{ij}=1，否则 c_{ij}=0
VM_i^{CPU}, VM_i^{MEM}	虚拟机 VM_i 的资源需求情况
$PM_{avg}^{CPU}, PM_{avg}^{MEM}$	CPU、内存的平均使用率
c_j^{CPU}, c_j^{MEM}	PM_j 拥有的全部资源情况

1) 安全性目标

根据 3.2 节中假设，恶意攻击者已知虚拟机放置策略，恶意攻击者可以通过不同的账户启动大量虚拟机来间接削弱 SVMPP 对用户间同驻次数的限制，最终完成同驻攻击。而 SVMPP 通过减少每台物理机上的用户数量来提升虚拟机安全性，以减轻联合攻击的危害，安全性度量由式(1)给出。

$$S:\min\frac{1}{|M_o|}\sum_{j=1}^{m}\sum_{i=1}^{h}c_{ij} \tag{1}$$

2) 负载均衡目标

负载均衡的重要性包含 2 个方面[20]。对于云服务提供商，将虚拟机均匀地分配到所有的物理机上可以减少单个物理机过度使用情况的发生。用户更希望将自己的多台虚拟机放置在不同的物理机上，既可以提高虚拟机的可用性，也能提高其容灾能力。

超额预定是云环境下的一种常见做法，它允许云服务提供商为用户分配比服务器实际拥有资源更多的资源，而 CPU 作为系统的关键资源始终启用超额预定，这种做法会使 RAM 成为系统资源的瓶颈。因此，选择 RAM 使用率作为负载均衡的度量标准。

假设物理机 PM_j 的内存使用率 $PM_j^{MEM}=VM_i^{MEM}\frac{a_{ij}}{c_j^{MEM}}$，当前组内物理机的平均内存使用率为 PM_{avg}^{MEM}，所有活跃物理机的负载均衡的方差为

$$L:\min\sqrt{\frac{\sum_{i=1}^{m}(PM_j^{MEM}-PM_{avg}^{MEM})^2}{|M_o|}} \tag{2}$$

3) 资源浪费目标

云环境下资源浪费的研究分为 2 个方面。1) 虚拟机放置过程的资源浪费问题。金顺福等[21]通过融合双速率和工作休眠的虚拟机调度策略，在保证系统响应性能的前提下降低了系统的空闲能耗。2) 虚拟机迁移过程中的资源浪费问题。李湘等[22]通过灰色预测模型对内存脏页面进行预测，之后根据脏页数对网络带宽环境进行预留调整，有效提升了网络传输速率，减少了迁移时间。崔勇等[23]使用基于博弈论的方法，公平有效地分配带宽，提高了多虚拟机迁移的性能。

为便于讨论，从 CPU 利用率及 RAM 利用率考虑资源浪费问题。假设一台物理机 PM_j 拥有的资源为 (c_j^{CPU}, c_j^{MEM})，一台虚拟机 VM_i 的资源需求为 (VM_i^{CPU}, VM_i^{MEM})，则物理机 PM_j 的 CPU 浪费率为 $\frac{C_j^{CPU} - \sum_{i=1}^{r} VM_i^{CPU}\alpha_{ij}}{C_j^{CPU}}$，该物理机的内存浪费率为 $\frac{C_j^{MEM} - \sum_{i=1}^{r} VM_i^{MEM}\alpha_{ij}}{C_j^{MEM}}$，资源浪费为

$$W : \min \sum_{i=1}^{m} W_j$$

$$W_j = \frac{C_j^{CPU} - \sum_{i=1}^{r} VM_i^{CPU} a_{ij}}{C_j^{CPU}} - \frac{C_j^{MEM} - \sum_{i=1}^{r} VM_i^{MEM} a_{ij}}{C_j^{MEM}}$$

$$VM_i^{CPU}\alpha_{ij} < C_j^{CPU},\ VM_i^{MEM}\alpha_{ij} < C_j^{MEM} \tag{3}$$

由于多目标优化模型的求解较为复杂，为便于快速求解，本文采用启发式算法中的线性加权和法，将多目标优化问题转换为单目标优化问题进行求解。引入 α、β 和 γ 分别作为安全性、资源浪费和负载均衡的权重系数，其中 $\alpha+\beta+\gamma=1$。加权处理后的优化目标为

$$F = \alpha S + \beta W + \gamma L \tag{4}$$

约束条件为

$$VM_i^{CPU}\alpha_{ij} < C_j^{CPU}$$

$$VM_i^{MEM}\alpha_{ij} < C_j^{MEM}$$

其中，$\forall i \in \{1,2,\cdots,N\}, \forall j \in \{1,2,\cdots,M\}$。

3.5 SVMPP 流程描述

SVMPP 将虚拟机的放置过程分为 4 步。

Step1 虚拟机的创建。用户 U 根据 Shamir 秘密共享方案将要保护的私密信息分为 n 份，将 n 个虚拟机创建请求发送至数据中心，并将相应门限值 k_u 一并发送。本文不关注用户如何进行秘密的分割及还原，只给出可能的解决办法[24]。

Step2 选取组管理者。这个过程根据用户限制表（UCT, user-constraint table）进行选取。UCT 记录用户所使用过的组管理者的记录，数据中心根据 UCT 选择组管理者并将虚拟机请求发送至相应的组管理者处，如表 2 所示。

表 2 用户限制表（UCT）

用户	使用过的组管理者
U_1	$\{G_1,G_2\}$
U_2	$\{G_2,G_3\}$
U_3	$\{G_1\}$
U_4	$\{G_1,G_2,G_4\}$

Step3 虚拟机放置。同驻限制表（CRT, co-resident table）记录着用户间的同驻限制，如表 3 所示。组管理者根据同驻限制表在本组内为用户选择合适的物理机，表 3 中 Pair 列记录 2 个用户及他们所在的组，Times 列记录 2 个用户在当前组内的同驻次数 $0 \leqslant t_i \leqslant K_i$，Threshold 列记录 2 个用户间同驻的最大次数 $2 \leqslant K_i$。组管理者收到虚拟机请求后，根据请求的 k_u 值与组内其他用户 k 值比较，选取较小的值作为 2 个用户间的同驻限制，更新 CRT。之后，组管理者根据 CRT 筛选符合条件的物理机，再筛选出满足资源条件的物理机，最后根据多目标优化选择最合适的物理机接受虚拟机请求，具体如算法 1 所示。若未找到合适的物理机，则撤销更新 CRT，并返回 Step1。

表 3 同驻限制表（CRT）

Pair	Times	Threshold
$<U_1,U_2,G_1>$	t_1	K_1
$<U_2,U_4,G_2>$	t_2	K_2
$<U_1,U_2,G_3>$	t_3	K_3

Step4 开启新的组。若当前开启的组内无法找到满足条件的物理机，则开启新的组管理者来接收当前虚拟机请求。

算法 1 SVMPP

输入 用户 U_i 的虚拟机序列 VM[N]，同驻限制表 CRT，用户限制表 UCT

输出 使用的服务器序列 Chosen_Server[N]，更新后的 CRT 及 UCT

1) begin

2) 从 UCT 中为用户选择未使用过的组管理者 GM[i]

3) min_F init to max

4) for each VM in VM[N]

5) for each Server in GM[i]

6) if CRT.append(VM,Server) < CRT Constraint then

7)　　$F \leftarrow \alpha\Delta S+\beta\Delta W+\gamma\Delta L$

8)　　if $F <$ min_F then

9)　　min_$F \leftarrow F$

10)　　将服务器添加到使用的服务器序列 Chosen_server 中

11) 返回 Chosen_server

12) 更新 CRT and UCT

13) end

3.6　算法复杂度分析

算法 1 的外层循环遍历所有虚拟机创建请求，时间复杂度为 O(M)，其中 M 为虚拟机数量。内层循环遍历组内的所有物理机，为虚拟机寻找同时满足 CRT 限制与资源限制的物理机，并根据多目标优化条件选择放置后效果最好的物理机，时间复杂度为 O(N)，其中 N 为组内的物理机数量。最终，算法执行时为每个虚拟机遍历一次组内的全部物理机，一共执行 mn 次，故算法整体的时间复杂度为 O(mn)。

3.7　两级管理框架中使用区块链

两级管理框架中的组管理者解决了数据中心可能成为云平台效率瓶颈的问题，但由于其维护的 Shamir 门限记录的重要性，使其易成为攻击者目标。一旦组管理者维护的 Shamir 门限记录被恶意篡改，所有运行在该组的虚拟机都将处于危险之中。为此，SVMPP 采用区块链技术保护记录不被篡改。下面首先介绍区块链的有关概念。

区块链是一个将区块按照时间顺序链接起来的一种链式数据结构。每个区块包含 2 个部分：header，存储上一区块的散列值；body，包含验证了区块创建过程中的交易记录。区块间通过 header 中存储的上一区块的散列值链接在一起，如图 3 所示。区块链是不断增长的，每当有区块被添加进区块链时，整条链便会向后延长，这个过程是由矿工完成的。矿工将当前交易记录写入区块，并通过大量的计算获取符合条件的散列值，最终获得在区块链中添加区块的权利，这个过程被称为矿工的挖矿过程。

区块链使用过程涉及的一个关键问题是矿工的选取问题。如图 4 所示，根据“击鼓传花”的规则在组管理者间选取矿工。设定一个令牌，令牌在组管理者间依次传递，当有交易发生时，持令牌者当矿工，由该矿工将交易记录写入区块，并将新区块添加进区块链。需要注意的是，若本次持令牌的矿工是交易记录中的参与者，则将令牌传递给下一个组管理者。这避免了某组管理者当矿工时，将自己参与的交易记录写入区块。

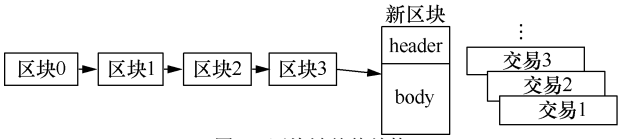


图 3　区块链整体结构

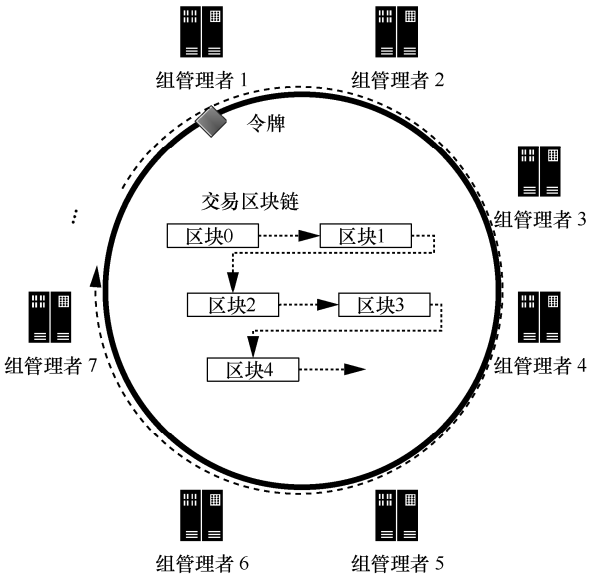


图 4　矿工选取过程

为激励矿工积极挖矿，对于不积极的矿工，取消其在下一周期内挖矿的资格。具体措施如下：在系统开始的一段时间内（几个周期）统计矿工完成挖矿所用的平均时间，记为 T_i。之后每隔几个周期重新统计并更新 T_i。在 T_i 的上一次更新后至下次更新前的周期内，将矿工当前完成挖矿时间 T_c 与相应的 T_i 进行比较，若 $T_c \leqslant T_i$，则保留该矿工在下个周期的挖矿资格，若 $T_c \geqslant T_i$，则取消该矿工下个周期的挖矿资格。本周期内没有挖矿资格的矿工将在下一周期自动恢复挖矿资格。

3.8　区块链的具体实现过程

由于 SVMPP 通过 CRT 与 UCT 来保障用户虚拟机安全性，如果 CRT 与 UCT 仅由单一的管理者维护，一旦该管理者被攻破，SVMPP 将无法继续有效地保障用户虚拟机的安全。为此，SVMPP 通过区块链技术来保障 CRT 与 UCT 的安全性。

数据中心产生 UCT 的交易记录，组管理者产生 CRT 的交易记录，记录产生后广播至其余所有的管理者处。组管理者与数据中心检查交易记录的有效性并返回该管理者的检查结果，并通过 Receive

函数接收其余管理者的认定结果，半数以上认同则 Receive 函数返回 True，将交易记录放入矿池。具体实现过程如下。

交易记录产生

1) begin
2) if UCT 更新
3) 创建关于 UCT 的交易记录
4) end if
5) if CRT 更新
6) 创建关于 CRT 的交易记录
7) 广播交易记录至所有组管理者
8) 接收组管理者的交易确认信息
9) /*交易信息确认则返回 True，否则返回 False*/
10) 半数以上的组管理者返回 True
11) 将交易记录放入矿池中
12) else
13) 丢弃该交易记录
14) end if
15)end

挖矿过程

1) begin
2) for 任意组管理者
3) if 该管理者持有挖矿令牌且 GMToken 为 True
4) 执行挖矿过程
5) else
6) 初始化该 GMToken 为 True
7) end if
8) if 矿池中含有该组管理者的交易记录
9) 将令牌传递至下一组管理者
10) end if
11) 对于矿池中的交易记录
12) 根据交易记录更新 UCT、CRT 并添加区块链
13) 令 T_{avg} 为该组管理者上次挖矿的时间
14) if 该组管理者本轮挖矿过程的时间 $T>T_{avg}$
15) 将该组管理者的挖矿令牌置为 False
16) 更新 T_{avg} 为本轮挖矿时间
17) end if
18) end for
19) end

4 实验及分析

为验证 SVMPP 在提高虚拟机安全性、提高负载均衡和减少资源浪费的有效性，使用 CloudSim[25-26]模拟实现了 SVMPP 放置策略，通过与 FirstFit、Random、CLR 放置策略对比，说明 SVMPP 在提高虚拟机安全性上的有效性。

此外，为直观描述 SVMPP 提高虚拟机安全性的有效性，引入同驻覆盖率，假设恶意用户 Mal 的目标用户 U 启动的虚拟机集合为 VIC={Vic_1, Vic_2, Vic_3,⋯,Vic_n}，恶意用户为达成同驻启动的虚拟机数量 Mal(VM)，同驻覆盖率（Co_Rate）定义为

$$\text{Co\_Rate}=\frac{|\text{Mal(VM)VIC}|}{|\text{VIC}|}$$

4.1 实验环境

本次实验设置如下：一个数据中心，多个组管理者，每个组管理者中都包含相同台数的物理机，物理机有 2 种配置（如表 2 所示）。为了提高实验的可信性，每个用户启动的虚拟机都从表 4 的 4 种配置中随机选择。

表 4 实验设置

	CPU 速率/MIPS	CPU 内核	RAM/MB
服务器	2 600	16	24 576
	2 600	12	49 152
	2 500	1	2 048
虚拟机	2 000	1	4 096
	1 000	1	8 192
	500	1	4 096

同时，为了实验方便，取所有用户的(k,n)=(4,10)，取多目标优化权重(α, β, γ)=（0.6,0.2,0.2）。在实际应用中，云服务提供商可以根据不同的(k,n)将服务器进行分组，相同限制的虚拟机请求发给特定类型的组管理者，在提高安全性的同时可以提升云环境负载均衡能力，降低资源浪费。权重系数按照云服务提供商对 3 个优化目标重要性的不同程度进行赋值。

4.2 实验结果及分析

为验证 SVMPP 提升虚拟机安全性的有效性，实验分别在 2 种物理机环境及不同分组大小下进行。

服务器配置 1：每个服务器的 CPU 内核数为 16，内存大小为 24 GB。

服务器配置 2：每个服务器的 CPU 内核数为 16 或 12，相应的内存大小为 24 GB 或 48 GB。

图 5 和图 6 分别在不同的配置环境下对 3 种分配策略下恶意用户的同驻覆盖率进行了比较，每组实验重复 30 次取平均值。横轴表示攻击者启动的虚拟机数量，纵轴表示同驻覆盖率。实验结果表明，SVMPP 下的同驻覆盖率要比 FirstFit 低约 34%，比 Random 低约 24%。

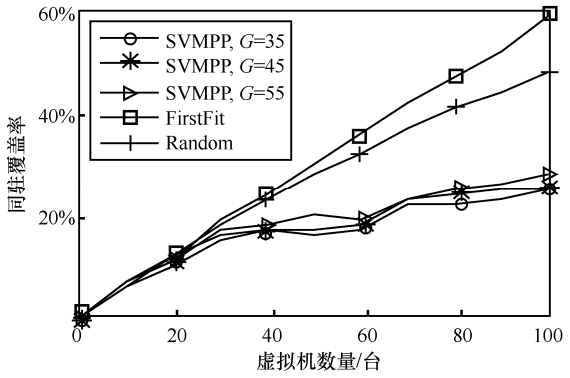


图 5　SVMPP 与现有放置策略安全性比较（服务器配置 1）

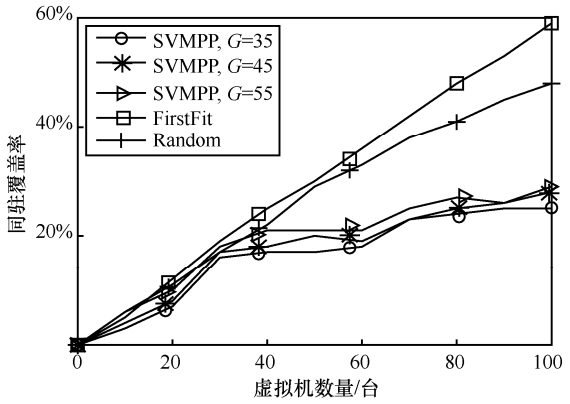


图 6　SVMPP 与现有放置策略安全性比较（服务器配置 2）

在 FirstFit 与 Random 策略下，随着恶意用户启动的虚拟机数量的增加，同驻覆盖率近似呈线性递增趋势，这是由于上述 2 种策略不对用户间的同驻次数进行限制，恶意用户启动的虚拟机数量越多，与目标的同驻概率越高，同驻覆盖率就越高。在 SVMPP 策略下，2 个用户间的同驻次数受 CRT 的限制，实验设定$(k,n)=(4,10)$，即用户间的同驻次数不超过 3，同驻覆盖率不超过 30%。并且，由于 CRT 与 Shamir 秘密共享方案的保护，攻击者无法获取足够的信息用于原始秘密的恢复，从而保护了原始秘密。

需要注意的是，在 2 种物理机配置下，SVMPP 的同驻覆盖率在达到 20%（同驻 2 次）左右开始明显平缓，这是由于 2 个用户间的同驻次数接近门限值（3 次）前，在组内找到同时满足门限需求与资源需求的物理机变得更加困难，所以同驻概率降低，导致同驻覆盖率降低。

与未考虑安全性的放置策略相比，SVMPP 安全性有较大的提升。为了保证实验的完整性，将 SVMPP（服务器配置 1）与 CLR 放置策略比较，在 CLR 策略实验中使用 150 台物理机，如文献[8]所述，当 N_{open} 越大时，抵御同驻攻击的效果越好，实验中也证明了这一点。SVMPP 与 CLR 安全性比较如图 7 所示。由图 7 可知，SVMPP 下同驻覆盖率要比 CLR 低至少 13%。这是由于 CLR 放置策略本质上类似随机放置策略，在状态为打开的虚拟机中随机选取物理机，当所有物理机状态均为打开时就是随机放置策略，这种状态也是 CLR 策略安全性最高的状态。此时 CLR 不对用户间的同驻次数进行限制，导致恶意用户通过启动大量的虚拟机依然可以达到较高的同驻覆盖率，而 SVMPP 策略通过 CRT 限制用户间的同驻次数，使恶意用户与目标的同驻覆盖率始终被限制在用户设定的门限内，提高了虚拟机的安全性。

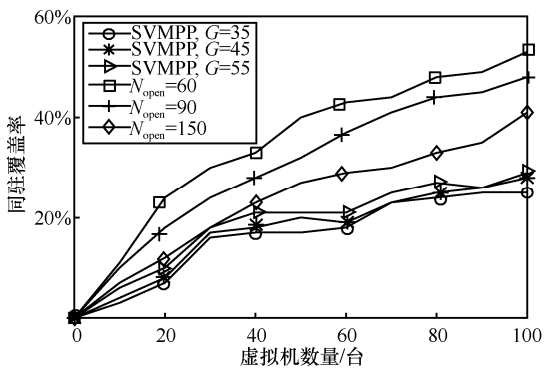


图 7　SVMPP 与 CLR 安全性比较（服务器配置 1）

同时，根据排队论的思想对 CLR 放置策略与 SVMPP 放置策略（组管理者数量为 5，G=55）下的虚拟机请求处理与等待时间做了对比，CLR 放置策略类似于 $M/M/1$ 排队模型，系统中仅数据中心处理虚拟机的放置，而 SVMPP 有多个组管理者同时处理放置任务，类似于 C 个 $M/M/1$ 的排队模型，实验结果如下。

图 8 展示了 2 个放置策略在不同虚拟机请求到达率下的平均响应时间，横轴表示每毫秒到达的虚拟机请求数。实验结果表明，在虚拟机请求到达率较小的情况下，SVMPP 策略下的虚拟机请求的平均响应时间稍高于 CLR 下的平均响应时间，这是由于 SVMPP 要进行查表操作，导致其单位时间的

处理能力要低于 CLR 策略，从而使平均响应时间高于 CLR 下的响应时间。虚拟机请求到达率增加后，由于 SVMPP 使用多个组管理者共同处理虚拟机放置任务，故处理时间无明显上升，而 CLR 由于仅依靠数据中心处理虚拟机放置，导致虚拟机请求的平均响应时间迅速增加。由此可知，SVMPP 在任务量大时有更好的处理能力。

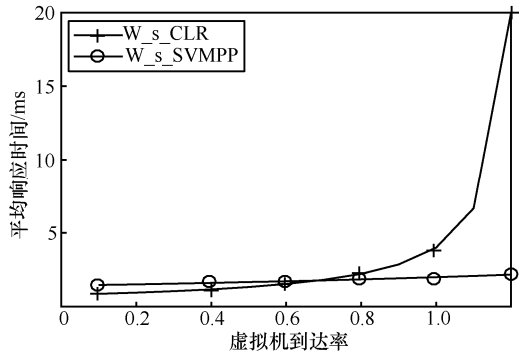


图 8 SVMPP 与 CLR 下虚拟机请求的平均响应时间

为描述 SVMPP 在负载均衡与资源浪费方面的有效性，实验使用 20 000 台虚拟机，根据所有开启的物理机中 RAM 使用率不同的物理机占比来衡量整体的负载均衡。

图 9 展示了 3 种分组大小下，内存利用率较高（大于 80%）的物理机占比。实验结果表明，当分组较小时，各用户间的同驻次数更容易达到阈值。这是由于每组的资源数量较少，当一部分用户进入该组后，会导致用户间的同驻次数迅速增长。尽管这时有一部分物理机还处于空闲状态，但已无法在该组为新用户找到合适的物理机，这种情况下会有部分物理资源被浪费，使整体的负载均衡效果受到影响。随着分组的增大，组内资源增多。在组内资源少的情况下（分组较小），被放置在 K+1 组的虚拟机在每组资源增加后可能会被放置在第 K 组，从而提升了第 K 组的资源利用率，使负载均衡的效果更理想。实验进一步讨论了在虚拟机数量一定的条件下，不同分组大小对负载均衡效果的影响。图 10 表示当虚拟机数量为 20 000 台时，不同分组大小下负载均衡的效果。从图 10 中可以看出，当分组大小为 40 时，负载均衡效果开始稳定，此时继续增加分组大小也无法提升负载均衡效果。

资源浪费是第三个优化目标，目的是减少内存利用率低的物理机数量，实验分别在不同分组大小下进行，每组实验重复 30 次取平均值。

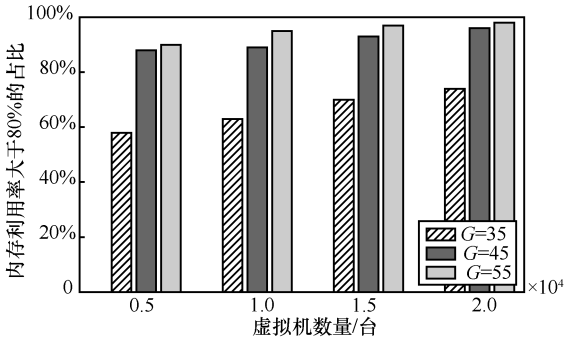


图 9 SVMPP 不同分组大小下负载均衡情况

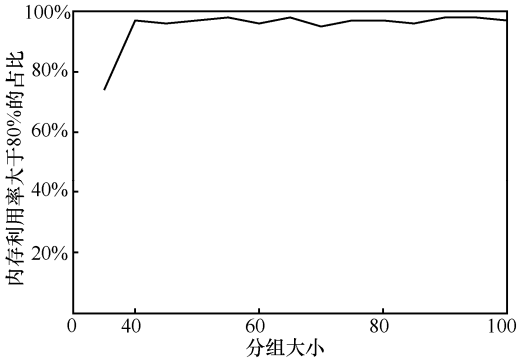


图 10 不同分组大小对负载均衡的影响（服务器配置 1）

图 11 展示了不同分组大小下，内存利用率较低（小于 80%）的物理机数量占总物理机数量的比例。实验结果表明，增大分组大小可以有效提升资源的利用率，这也与之前负载均衡的结论一致。

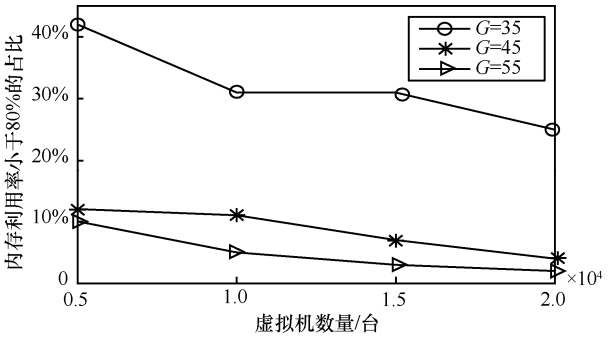


图 11 SVMPP 不同分组大小下资源使用情况（服务器配置 1）

实际中对 Shamir 条件在(k,n)=(5,10)情况下的安全性、负载均衡和资源浪费也进行了测试，实验结果表明，同驻次数对优化的 3 个目标均有影响，同驻限制次数的增加意味着安全性的降低，但是可以提升负载均衡的效果，提高资源的使用率。由于篇幅限制，在此不再详述。

4.3 SVMPP 的实用性

国内云服务提供商服务器价格如表 5 所示。由

表 5　　**国内云服务提供商服务器价格（截止到 2018 年 6 月 25 日）**

云服务提供商	实例类型	核心数	内存/GB	带宽/(Gbit·s^{-1})	价格/(元·月$^{-1}$)	API 名称
腾讯云	小型	4	8	1.5	364	S3.Large8
	大型	8	16	1.5	728	S3.Large16
阿里云	小型	2	4	2	179	ecs.c5.large
	中型	4	8	1.5	358	ecs.c5xlarge
	大型	8	16	2.5	716	ecs.c5.2xlarge

表 5 可知，将秘密分成多份放置在多个虚拟机中并不会带来过多的额外开销。比起虚拟机被侧信道攻击，云租户将愿意承担由于将计算从少数大型 VM 转移到众多小型 VM 而导致的成本。

综上，SVMPP 有效地提升了虚拟机的安全性和云环境整体的负载均衡能力，减少了资源浪费。

5　结束语

本文提出了一种新的虚拟机放置策略，通过用户与 CSP 两方面努力提升虚拟机的安全性，并且通过多目标优化对负载均衡和资源浪费进行了优化。设计了虚拟机放置的两级管理模型，一方面，满足了用户需要多次保护不同秘密的需求；另一方面，在两级管理框架中通过区块链保障了放置策略中关键数据的安全性。

未来将考虑如下几个问题：1)多数据中心，本文的虚拟机放置策略只考虑了单个数据中心下的虚拟机放置问题，多数据中心下的虚拟机放置问题还有待研究；2) 虚拟机动态迁移，本文的虚拟机放置策略是在虚拟机的初始化放置阶段，动态迁移有利于进一步提高虚拟机的安全性，也更有利于平衡负载均衡，减少资源浪费。

参考文献：

[1] RISTENPART T, TROMER E, SHACHAM H, et al. Hey, you, get off of my cloud:exploring information leakage in third-party compute clouds[C]//ACM Conference on Computer and Communications Security. 2009: 199-112.

[2] ZHANG Y, JUELS A, REITER M K, et al. Cross-VM side channels and their use to extract private keys[C]//ACM Conference on Computer and Communications Security. 2012: 305-316.

[3] LIU F, YAROM Y, GE Q, et al. Last-level cache side-channel attacks are practical[C]//IEEE Symposium on Security and Privacy. 2015: 605-622.

[4] YOUNIS Y A, KIFAYAT K, SHI Q, et al. A new prime and probe cache side-channel attack for cloud computing[C]//The 13th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC-2015). 2015:1718-1724.

[5] INCI M S, GÜLMEZOGLU B, APECECHEA G I, et al. Seriously, get off my cloud! Cross-VM RSA key recovery in a public cloud[J]. IACR Cryptology ePrint Archive, 2015, 2015: 1-15.

[6] CHEN K, SHEN Q, LI C, et al. Sift - an efficient method for co-residency detection on Amazon EC2[C]//International Conference on Information Systems Security and Privacy. 2016: 423-431.

[7] BATES A, MOOD B, PLETCHER J, et al. On detecting co-resident cloud instances using network flow watermarking techniques[J]. International Journal of Information Security, 2014, 13(2): 171-189.

[8] ZHANG Y, LI M, BAI K, et al. Incentive compatible moving target defense against VM-colocation attacks in clouds[C]//IFIP International Information Security Conference. Springer, 2012: 388-399.

[9] AZAR Y, KAMARA S, MENACHE I, et al. Co-location-resistant clouds[C]//The 6th Edition of the ACM Workshop on Cloud Computing Security. 2014: 9-20.

[10] HAN Y, CHAN J, ALPCAN T, et al. Using virtual machine allocation policies to defend against co-resident attacks in cloud computing[J]. IEEE Transactions on Dependable & Secure Computing, 2017, 14(1): 95-108.

[11] AFOULKI Z, BOUSQUET A, ROUZAUD-CORNABAS J, et al. A security-aware scheduler for virtual machines on IaaS clouds[J]. Rapport de Recherche, 2011, 8: 1-12.

[12] BERRIMA M, NASR A K, BEN R N, et al. Co-location resistant strategy with full resources optimization[C]//The 2016 ACM on Cloud Computing Security Workshop. 2016: 3-10.

[13] BARROWCLOUGH J P, ASIF R. Securing cloud hypervisors: a survey of the threats, vulnerabilities, and countermeasures[J]. Security and Communication Networks, 2018 (2018): 1681908:1-1681908:20.

[14] NEZARAT A, SHAMS Y. A game theoretic-based distributed detection method for VM-to-hypervisor attacks in cloud environment[J]. Journal of Super Computing, 2017, 73(2): 1-21.

[15] WANG C, MA S, ZHANG X, et al. A hypervisor level provenance system to reconstruct attack story caused by kernel malware[C]//International Conference on Security and Privacy in Communication Systems. 2017: 778-792.

[16] LIU L, WANG A, ZANG W Y, et al. Empirical evaluation of the hypervisor scheduling on side channel attacks[C]//2018 IEEE International Conference on Communications. IEEE, 2018: 1-6.

[17] SZEFER J, KELLER E, LEE R B, et al. Eliminating the hypervisor attack surface for a more secure cloud[C]//The 18th ACM conference on Computer and Communications Security. ACM, 2011: 401-412.

[18] 石勇, 郭煜, 刘吉强, 等. 一种透明的可信云租户隔离机制研究[J]. 软件学报, 2016, 27(6): 1538-1548.
SHI Y, GUO W, LIU J Q, et al. Research on a transparent trusted cloud tenant isolation mechanism[J]. Journal of Software, 2016, 27(6):

1538-1548.

[19] VARADARAJAN V, ZHANG Y, RISTENPART T, et al. A placement vulnerability study in multi-tenant public clouds[C]//USENIX Security Symposium. 2015: 913-928.

[20] JANSEN R, BRENNER P R. Energy efficient virtual machine allocation in the cloud[C]//Green Computing Conference and Workshops. 2011: 1-8.

[21] 金顺福, 郝闪闪, 王宝帅. 融合双速率和工作休眠的虚拟机调度策略及参数优化[J]. 通信学报, 2017, 38(12): 10-20.
JIN S F, HAO S S, WANG B S. Virtual machine scheduling strategy based on dual-speed and work vacation mode and its parameter optimization[J]. Journal on Communications, 2017, 38(12): 10-20.

[22] 李湘, 陈宁江, 杨尚林, 等. 感知应用特征与网络带宽的虚拟机在线迁移优化策略[J]. 通信学报, 2017, 38(Z2): 147-155.
LI X, CHEN N J, YANG S L, et al. Optimization strategy of virtual machine online migration with awareness of application characteristics and network bandwidth migration[J]. Journal on Communications, 2017, 38(Z2): 147-155.

[23] 崔勇, 林予松, 李润知, 等. 基于合作博弈的多虚拟机实时迁移带宽分配机制[J]. 通信学报, 2016, 37(4): 149-158.
CUI Y, LIN Y S, LI R Z, et al. Cooperative game based bandwidth allocation mechanism live migration of multiple virtual machines[J]. Journal on Communications, 2016, 37(4): 149-158.

[24] 荣辉桂, 莫进侠, 常炳国, 等. 基于 Shamir 秘密共享的密钥分发与恢复算法[J]. 通信学报, 2015, 36(3): 64-73.
RONG H G, MO J X, CHANG B G, et al. Key distribution and recovery algorithm based on Shamir secret sharing [J]. Journal on Communications, 2015, 36(3): 64-73.

[25] CALHEIROS R N, RANJAN R, BELOGLAZOV A, et al. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms[J]. Software Practice & Experience, 2011, 41(1): 23-50.

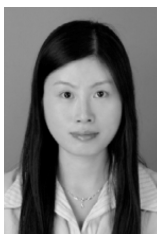
[26] ALBDOUR L. Comparative study for different provisioning policies for load balancing in cloudsim[J]. International Journal of Cloud Applications and Computing (IJCAC), 2017, 7(3): 76-86.

[作者简介]

田俊峰（1965− ），男，河北保定人，博士，河北大学教授、博士生导师，主要研究方向为信息安全与分布式计算。

王子龙（1995− ），男，河北石家庄人，河北大学硕士生，主要研究方向为云安全、同驻攻击等。

何欣枫（1976− ），男，天津人，河北大学博士生，主要研究方向为云计算安全、可信计算等。



李珍（1981− ），女，河北保定人，河北大学副教授，主要研究方向为软件安全、可信计算。