云存储中的 ORAM 研究综述

顾晨阳[1] 付 伟[1] 刘金龙[2] 孙 刚[2]
(海军工程大学信息安全系 武汉 430033)[1] (海军机要密码室 北京 100841)[2]

摘 要 在云存储环境中,服务器或者第三方可以仅通过对用户访问行为进行分析来获取信息,对用户信息安全造成威胁。ORAM 通过构造精巧设计的存储结构和冗余的访问机制,有效地隐藏用户访问行为与访问目标之间的对应关系。隐藏用户访问意图的安全访问机制,是现阶段隐藏用户访问模式的主要手段之一。通过对 ORAM 基本理论和发展历程进行研究,归纳分析了 ORAM 的基本方案;建立了 SSIBT 性能评价指标体系,对经典 ORAM 算法及其优化方案进行了分析比较;最后,在分析现阶段主要研究重点的基础上,总结提出了 ORAM 未来可能的主要研究方向。

关键词 云存储,ORAM,行为安全,访问机制,隐私保护
中图法分类号 TP309 **文献标识码** A

Survey of ORAM Research in Cloud Storage

GU Chen-yang[1] FU Wei[1] LIU Jin-long[2] SUN Gang[2]
(Department of Information Security, Naval University of Engineering, Wuhan 430033, China)[1]
(Naval Confidential Password Office, Beijing 100841, China)[2]

Abstract In a cloud storage environment, servers and the third party can fetch information through analyzing the users' access behaviour, which may cause threats to users' information security. ORAM mechanism is one of the main strategies which can hide users' visiting patterns. This mechanism can effectively conceal the corresponding relationships between the access behaviour and the visiting targets. Secure access mechainsm to hide user's access intention is one of the main means to hide user's access model at present. Through the study of the basic theories and the development process of the ORAM, this paper concluded the basic scheme of this mechanism and set up a SSIBT performance evaluation index system to make comparisons and analysis between the classic ORAM mechanism and its optimization scheme. Finally, possible research directions of ORAM were summarized based on the main research focus.

Keywords Cloud storage, ORAM, Behavioral security, Access mechanisms, Privacy protection

1 引言

在云存储环境中,用户把数据提交给云服务器,由云服务器存储和管理数据。用户只拥有数据的所有权,服务提供商拥有数据的直接控制权,而且用户无法确保服务提供商是否按照协议保证数据存储的安全性。如果用户以明文的形式将数据提交给云服务器,那么作为不可信的服务提供商就能直接获取用户的数据。传统的应对方法是通过加密来保护数据安全,用户提交的数据和交互的信息均为密文信息。但是,密文存储[1-3]仍然存在安全隐患,加密只能保证数据内容本身不被泄露[4],服务提供商可以通过对用户远程访问方式的记录和观察开展行为特征分析与处理[5],可以挖掘出用户的敏感信息或者对解密有用的信息。用户的访问行为主要指用户访问数据的频度、顺序、关联等。例如,某医生在流感高发季高频率查询某些资料,可以推断该资料可能与流感有关。因此,如何隐藏用户的访问行为,是目前研究的重点和难点问题之一。

不经意随机访问机(Oblivious Random Access Machine, ORAM)[6]通过构造恰当的存储结构和访问机制,有效隐藏用户访问行为与访问目标之间的对应关系,是隐藏用户访问意图的安全访问机制。ORAM 是现阶段隐藏用户访问模式的主要手段。用户访问行为主要分为读操作和写操作[7],通过对读写操作的变换增加一些冗余操作。攻击者无法得知哪些操作是有效的,从而无法根据用户的行为来获取有效信息。用户上传存储在服务器的数据,包括有效数据和冗余数据,以混淆用户访问的目标数据,从而隐藏用户的访问行为。研究表明:ORAM 在云存储环境下能够有效保护用户数据,防止第三方根据用户的访问行为来获取数据信息,在单纯依靠加密技术来保护数据的传统框架下,为数据增加了一道安全机制,提高了云端数据的安全性[8]。但是,ORAM 在提高安全性的同时,会对服务器造成一些额外的负担。例如:为混淆目标数据块,增加部分冗余数据块,对不同数据块进行多次访问,增加存储空间负担和交互数据量。如何在保证用户访问行为安全性的前提下提高 ORAM 的访问效率是一个挑战性问题。

本文受国家自然科学基金项目(61672531),总装后勤科技重大项目子课题(AWS14R013)资助。
顾晨阳(1995—),男,硕士生,主要研究方向为云存储安全、信息安全;**付 伟**(1978—),男,博士,副教授,CCF 会员,主要研究方向为云计算、云安全、分布式计算、信息安全,E-mail:lukeyoyo@tom.com(通信作者)。

本文第 1 节主要介绍了云存储的发展和 ORAM 产生的基本背景;第 2 节对 ORAM 的定义、ORAM 安全性、基本操作、基本特性和 ORAM 的发展历程进行概述;第 3 节对现阶段 ORAM 的研究成果进行概述;第 4 节对针对 ORAM 的优化方案和优化策略进行概述;第 5 节建立 ORAM 评价体系,并对现阶段 ORAM 方案的性能进行分析比较;第 6 节提出了 ORAM 技术未来发展的主要方向;最后对本文工作进行总结。

2 ORAM 基本理论

2.1 ORAM 定义

ORAM 从 RAM(Random Access Machine)模型演变而来。RAM 中的处理器通过对存储器进行读写操作来实现功能[9],是计算仿真的重要手段。为了防止攻击者通过获取程序对内存的访问模式信息[10],实现软件的逆向工程,Goldriche 和 Ostrovsky 在 1996 年首先提出了 ORAM,用于保护用户的访问模式。访问模式,是指处理器访问存储器内部数据的操作序列和地址序列[11]。ORAM 采用两种技术实现对访问模式的保护。一是通过修改物理地址保护访问的地址序列。用户在对任意数据块进行操作后,ORAM 将更改受访问数据块的物理地址,将数据块写回新的物理地址,确保用户的任意两次访问不会产生地址关联。二是通过重加密保护操作序列。用户在写回数据块操作前,都需要对新数据块进行重加密操作[12],以确保攻击者不能通过读写密文的内容的异同判断是读操作还是写操作。

2.2 ORAM 安全性

ORAM 的主要功能是通过将简单的访问操作复杂化,混淆访问目标和操作,从而实现对用户访问行为的安全保护[13],其安全性可以用定义 1 加以描述。

定义 1(ORAM 安全[14]) $y=(op,\mathrm{arg})$表示用户的一次访问操作。其中 op 表示用户的操作类型,包括读操作(read)或者写操作(write);arg 表示操作对象,即目标数据块。用户的两次不同访问请求为 $y=(op_1,arg_1)$和 $z=(op2,arg_2)$,通过 ORAM 技术变换为访问行为 $oram(y)$和 $oram(z)$。如果在计算上 $oram(y)=oram(z)$,则称一个 ORAM 是安全的。

亦即,从第三方角度来看这两次访问请求是一致的,第三方无法从中获取任何与访问有关的信息。在基础的 Trivial ORAM 中,用户按序访问每一个数据块,$oram(y)=(op(data_1),op(data_2),op(data_3)\cdots)$,$oram(z)=(op(data_1),op(data_2),op(data_3)\cdots)$,在计算上 $oram(y)=oram(z)$。对于不同的访问请求,从访问行为上看是一致的,第三方无法从中获取有效信息,可以证明 ORAM 是安全的。

2.3 ORAM 基本操作

为保证用户的访问行为在计算上不可区分,ORAM 将用户的每一次访问操作细化成读取、解密、读写、重加密、写回这 5 个基本操作[15]。

为了保证存储数据的安全性,在进行一定次数的访问后,需要进行重排操作[16]。将数据块打乱,重新分配数据块的存储位置,避免第三方通过数据的位置信息分析获取相关隐私信息。

在二叉树存储模型中,每个存储单元为 bucket,存储的数据大小有限,需要不断进行驱逐操作,将满 bucket 中的数据块移动到空闲的 bucket,避免发生数据溢出。

2.4 ORAM 基本特性

ORAM 能有效保护用户的访问行为,防止信息泄露,是云存储安全技术的重要部分。ORAM 主要有以下 4 个特性。

(1)安全性:在采用加密的密文存储技术[17]保证数据内容安全的基础上,进一步保证数据的位置信息安全和访问行为安全。

(2)广泛性:能在用户读写密文数据时保护用户的访问模式,在密文搜索、软件逆向工程、安全计算等多个领域的安全性保障中起着至关重要的作用。

(3)冗余性:为保护云服务器存储的数据块安全,增加了大量冗余的无效数据块,这增加了数据存储空间的开销。

(4)低效性:将直接的访问操作细化,通过额外的操作保护访问行为的安全,同时,增加的无效数据块会降低搜索等操作的效率。

如何在保证安全性的同时提高存储访问的效率,是目前该领域研究的重点和热点问题之一。

2.5 ORAM 的发展历程

随着云计算的迅速发展和数据的爆炸式增长,云存储应运而生,同时安全问题[18]也日趋严峻。1996 年 Ostrovsky 提出 ORAM,主要是解决软件保护[19]和防止软件逆向工程等攻击问题[20]。他提出的 Square Root ORAM 和 Hierarchical ORAM 两种方案能够应用到云存储环境中,有效保护用户对云服务器中数据的访问模式。伴随着数据量的不断增加,这两种方案数据访问复杂、算法实现复杂的缺点逐渐暴露。Stefanov[21]在原有方案上进行优化,提出 Partition ORAM,其采用新的划分方法提高访问效率,但是没有从根本上解决访问效率低的问题。

Shi 等[22]于 2011 年创新性地提出在云服务器以二叉树结构存储数据块,通过改变服务器的存储方式提高数据搜索、写回和访问的效率。在二叉树存储结构的基础上,ORAM 进入高速发展阶段,出现了 Tree ORAM,Path ORAM,Ring ORAM 等新型方案。

Doerner 等[23]于 2017 年创新性地提出了分布式 ORAM 概念,将读写操作隔离,分为只读存储(read-only memory)和只写存储(write-only memory)两部分,加快了数据读取的速度。同时,利用基于函数秘密共享的隐私信息检索,实现数据读写不暴露隐私信息的目标。

通过优化策略来提高效率,是现阶段 ORAM 研究的热点,尤其是在 2018 年,Xiao[24]等在二叉树存储结构的基础上,结合隐私路径检索技术,提出了一种基于两服务器的高效 ORAM 计算协议。通过隐私检索技术来获取目标数据块路径信息,取消了在每次访问后对地址映射表的更新操作,减少了交互轮数,提高了效率。

ORAM 作为防止访问模式泄露的关键技术,成为云存储数据安全研究的热点问题,得到了专家学者们的高度关注。如何在保证用户访问安全性的前提下设计实现高效的 ORAM 方案,是现阶段 ORAM 研究的主要方向。

3　ORAM 研究成果

自 ORAM 概念提出以来，研究人员从存储结构、算法操作、访问机制等方面对 ORAM 不断改进，提高用户的访问效率，形成高效且安全的 ORAM 方案。

3.1　经典 ORAM 方案

3.1.1　Trivial ORAM 方案

Trivial ORAM 是最基础的 ORAM。数据在云端以最简单的数组结构连续存储在服务器中，对用户访问行为不提供任何保护。Trivial ORAM 主要通过用户的访问机制来保护用户的访问行为。用户访问某一数据块时，需要依次遍历所有数据块。用户每次读取一个数据块，并对其进行解密匹配。若该数据块为目标数据块，则对该数据块进行读写操作和重加密操作，并将重新加密的数据块写回云端数据库。若该数据块不为目标数据块，则直接进行重加密和写回操作。无论是否为目标数据块，算法都将读取下一个数据块，直至遍历完所有数据块。

Trivial ORAM 主要有两个优点：1）将用户对目标数据块的访问请求转化为依次读取所有数据块并进行比较辨别，确保第三方无法确定用户访问的目标，从而保护了用户的访问行为；2）遍历算法和比较算法是基础算法，实现起来比较容易。但是，在保护用户服务行为的同时，牺牲了时间和空间效率。将对一个数据块的访问转化为所有数据块的读取比较，同时需要大量的缓冲空间，效率很低。

3.1.2　基于访问算法优化的 ORAM 方案

Trivial ORAM 访问目标数据块时需要读取所有数据块，因此在数据量较大的应用场景中效率极其低下。为提高访问效率，Ostrovsky 提出了 Square Root ORAM[7]，其基本思路是：在服务器增加缓冲区 shelter，减少用户每次访问需要搜索的数据块数量，从而提高访问效率；同时，在服务器增加无效数据块，进一步隐藏目标数据块，保证访问行为的安全。

服务器将用户提交的 N 个有效数据块和服务器随机产生的 $\sqrt{N}$ 个无效数据块以数组的形式存放，并开辟大小为 $\sqrt{N}$ 的缓冲区 shelter 作为数据调动存储和用户直接搜索的区域。Square Root ORAM 存储结构[25]如图 1 所示。

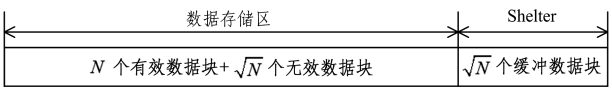

图 1　Square Root ORAM 存储结构

Square Root ORAM 在服务器内以数组的形式存储 N 个有效数据块、$\sqrt{N}$ 个无效数据块、$\sqrt{N}$ 个缓冲数据块，其中有效数据块和无效数据块是混合乱序存储。

Square Root ORAM 的主要步骤如下：

（1）用户向服务器提交访问需求；

（2）服务器访问缓冲区 shelter，遍历搜索目标数据块；

（3）若找到目标数据块，则从数据存储区随机读取一个无效数据块写入 shelter；若没有找到目标数据块，则从数据存储区内搜索目标数据块，找到后将目标数据块写入 shelter；

（4）用户遍历搜索 shelter，读取目标数据块，并进行读写操作；

（5）用户将新数据块写回 shelter，覆盖原数据块。

记 $\sqrt{N}$ 次用户访问为一个周期，每个周期结束后，将 shelter 内数据块写回数据存储区，将对应的原数据覆盖，最后将数据存储区内所有数据块乱序重组。

3.2　基于存储结构优化的 ORAM 方案

在 Trivial ORAM 方案中，云服务器数据以数组结构连续存储，在数据量较小的情况下能够较为快速地完成用户的访问请求。但是，在大数据的应用环境下，时间消耗太大，严重影响了用户体验。为提高访问效率，Rafail[7]提出将数组存储转化为哈希表存储。Shi 等[22]更是创新性地提出二叉树存储结构，开创了 ORAM 研究的新思路。

3.2.1　基于哈希表的 Hierarchical ORAM

Hierarchical ORAM 中的数据以哈希表的形式存放在服务器端的存储空间，建立层次存储模型。服务器以层次模型存储数据，每一层都是哈希表。每个哈希表都有一个与之对应的哈希函数 $hash_i$，用户可以通过 $hash_i$ 判断目标数据块是否在该行，计算目标数据块在哈希表的具体位置。为了避免哈希冲突，将数据块存储在大小为 $b=O(\log N)$ 的 bucket 中。Hierarchical ORAM 的数据结构[25]如图 2 所示。

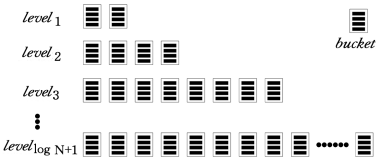

图 2　Hierarchical ORAM 数据结构

服务器每层都是哈希表，记为 $level_i$，共有 $\log N+1$ 层，以 bucket 为最小存储单元，第 i 层存储 2^i 个 bucket。

Hierarchical ORAM 的主要步骤如下：

（1）用户向服务器提交访问需求；

（2）服务器搜索第一层，判断目标数据块是否在该层，若在，则根据哈希函数的偏移量寻找目标 bucket 并提交给用户；若不在，则随机将该层一个包含无效数据块的 bucket 提交给用户；

（3）用户遍历服务器提交的 bucket，搜索目标数据块，若没有目标数据块，则不进行操作；若发现目标数据块，则进行读写操作，并更新数据块和 bucket；

（4）服务器按序搜索下一层，如果在上层没有找到目标数据块，则搜索该层，搜索方式与上层相同；如果已经在上层找到目标数据块，则随机将该层一个包含无效数据块的 bucket 提交给用户；

（5）服务器搜索完所有层；

（6）用户将新 bucket 返回给服务器；

（7）服务器将新的 bucket 写入第一层，如果第一层存在原 bucket，则覆盖替换原 bucket，否则覆盖替换任意一个 bucket，并更新哈希函数 $hash_1$。

对于第 i 层，以 $2i$ 为一个周期。当第 i 层一个周期结束后，服务器将该层所有的 bucket 与第 $i+1$ 层的 bucket 合并。如果存在对应某一数据块的两个 bucket，则保留时间戳较新的 bucket，删除旧的 bucket。将所有 bucket 混乱重排入第 $i+1$ 层，并为第 $i+1$ 层更新哈希函数 $hash_{i+1}$。

3.2.2　基于二叉树的 Tree ORAM

服务器以二叉树结构存储数据块，以 bucket 作为二叉树的节点，建立高度为 $h=\log N$ 的二叉树，共有 N 个叶子节点。服务器将数据块存储在二叉树的节点中，用该节点所在路径

的叶子节点表示 bucket 的存储地址，并在用户端存储一个数据块地址映射表，用于存储每个数据块的存储地址。Tree ORAM 的数据结构如图 3 所示。

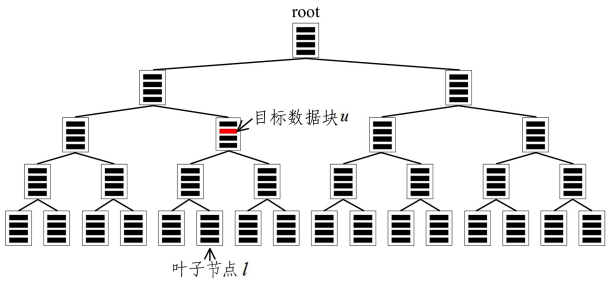

图 3 Tree ORAM 算法的数据结构(电子版为彩色)

图 3 中目标数据块 u 的地址为叶子结点 l，说明目标数据块存储在路径 $path(l)$ 上的某一个 bucket 中。路径 $path(l)$ 表示从根节点 root 到叶子节点 l 的路径，即图中绿色路径。

Tree ORAM 的主要步骤如下：

(1)用户向服务器提交访问需求；

(2)查询数据块地址映射表，确定目标数据块所在路径；

(3)遍历搜索 root 节点，找到目标数据块，则将目标数据块提交给用户；若未找到，则提交一个无效数据块；

(4)从 root 节点开始，沿着目标路径依次搜索每一个节点 bucket，若已在之前的节点 bucket 找到目标数据块，则在搜索后面的节点 bucket 时，直接向用户提交一个节点 bucket 中的无效数据块，直至叶子结点；

(5)用户对目标数据块进行操作，产生新数据块；

(6)为新数据块分配新的地址，并更新数据块地址映射表；

(7)将新数据块写回 root 节点 bucket 中。

由于每次写回数据都是写回 root 节点 bucket 中，而存储空间有限，因此需要进行回收操作。从 root 节点 bucket 开始，依次从每一层随机选择 x 个节点 bucket，若选中节点 bucket 中存在有效数据块，则随机选取一个有效数据块和一个无效数据块，若没有有效数据块，则随机选择两个无效数据块。将有效数据块移动到该数据块地址路径上的子节点 bucket，无效数据块则移动到另一个子节点 bucket。

3.3 基于驱逐操作优化的 ORAM 方案

在二叉树存储结构的 ORAM[22] 中，为保证每个 bucket 中的数据块不溢出，需要周期性地进行驱逐操作，这是影响访问效率的关键因素。

3.3.1 基于缓冲区写回的 Path ORAM

Path ORAM[26] 在客户端设置一个缓冲区 stash，用于存放数据块。通过查询数据块地址映射表确定目标数据块所在路径，然后将从 root 节点到目标叶子结点路径上的所有 bucket 数据块读取至 stash 中，用户直接从 stash 中搜索目标数据块。用户对目标数据块进行更新后，为新数据块分配新地址，同时更新数据块地址映射表，并将 stash 中的数据块尽可能多地写回服务器。为减少服务器存储空间，将 bucket 的容量规定为常量 Z，这在一定程度上减少了存储空间，提高了空间利用率。

Path ORAM 的主要步骤如下：

(1)用户向服务器提交访问需求；

(2)查询数据块地址映射表，确定目标数据块所在路径；

(3)将从 root 节点到目标叶子节点路径上的所有 bucket 中的数据块上传到 stash 中，此时 stash 中有新上传提交的数据块，也有之前操作没有写回服务器的数据块；

(4)用户遍历 stash，查找目标数据块；

(5)用户对目标数据块进行操作，产生新数据块；

(6)为新数据块分配新的地址，并更新数据块地址映射表；

(7)将新数据块写回 stash 中，覆盖原数据块；

(8)从 stash 尽可能多地随机选取数据块写回服务器中。

用户将 stash 中的数据块写回服务器时，从目标叶子结点开始，反序浏览读取数据块的原路径，将需要写回的数据块写入数据块路径和读取路径的第一个重合 bucket 中。Path ORAM 写回算法如图 4 所示。

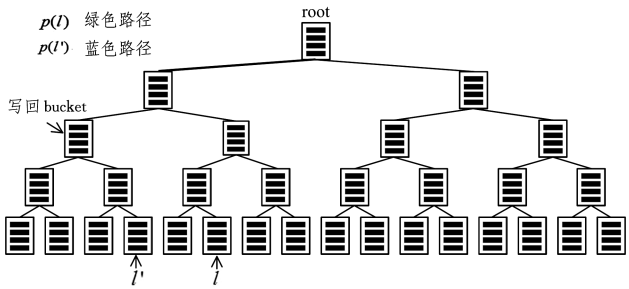

图 4 Path ORAM 写回算法

读取数据块 u 时，数据块 u 的地址即目标叶子结点为 l，读取路径为 $p(l)$。用户在 stash 中随机选取数据块 a 写回服务器，数据块 a 的地址为 l'，写回数据块路径为 $p(l')$，则反序浏览 $p(l)$，将数据块 a 写入与 $p(l')$ 重合的第一个 *bucket* 中。

3.3.2 基于 bucket 修正的 Ring ORAM

Ring ORAM[27-29] 是 Ren 在 2014 年提出的新型 ORAM 方案。Ring ORAM 中采用修正算法，每个 bucket 中的有效数据块有 X 个，无效数据块有 Y 个。bucket 中的有效数据块被读取至 stash 后，服务器会将 bucket 中的原数据块删除，由于 bucket 中的有效数据块只有 X 个，也就说，当对一个 bucket 进行 X 次读取操作后，需要对该 bucket 进行重组。

Ring ORAM 的主要步骤如下：

(1)用户向服务器提交访问需求；

(2)查询数据块地址映射表，确定目标数据块所在路径；

(3)遍历搜索 root 节点，找到目标数据块，则将目标数据块提交给用户；若未找到，则提交一个无效数据块；

(4)从 root 节点开始，沿着目标路径依次搜索每一个节点 bucket，若已在之前的节点 bucket 找到目标数据块，则在搜索后面的节点 bucket 时，直接向用户提交一个节点 bucket 中的无效数据块，直至叶子结点；

(5)若仍未找到目标数据块，则遍历 stash，查找目标数据块；

(6)用户对目标数据块进行读写操作，将操作后的数据块重新封装为新的数据块；

(7)为新数据块分配新的地址，并更新数据块地址映射表；

(8)将新数据块写回 stash 中。

为保证数据的茫然性，每进行 $O(X)$ 次访问操作，就要选择对服务器的某一个二叉树存储路径进行重组操作。重组操作时，将该路径上所有数据块读取到 stash 中，然后从该路径的叶子结点开始，把 stash 中的数据块写回，写回时要保证 bucket 中的数据块离自身的叶子结点最近。

3.4 基于访问机制改进的 ORAM 方案

Partition ORAM[21] 采用一种新颖的划分方法，将数据存

储量为 N 的存储空间划分为 $\sqrt{N}$ 个小型 ORAM，称其为子系统。将大型数据存储空间分解成多个小型数据存储空间，每一个小型数据存储空间在服务器端都采用 Square Root ORAM，Hierarchical ORAM 等 ORAM。同时，在用户端存储一个映射表，用于存储数据块与子系统的对应关系。对应关系表示数据块存储在对应子系统中的 ORAM 或者 stash 中。Partition ORAM 数据结构如图 5 所示。

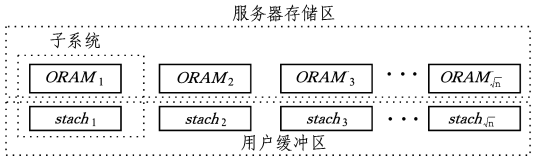

图 5　Partition ORAM 数据结构

将服务器储存空间划分为 $\sqrt{N}$ 个储存区域，在用户端分别建立缓冲区 $stash_i$ 对应不同的区域。

Partition ORAM 的主要步骤如下：

(1)用户查找映射表，确定目标数据块所在子系统；

(2)在用户端搜索目标数据块所在子系统的 stash，若未找到，则向服务器提交目标数据块的访问需求，服务器在对应的 ORAM 中进行搜索，找到后将目标数据块提交存储在对应的 stash 中；若在 stash 中找到，则向服务器提交一个无效数据块的访问需求，服务器将一个无效数据块提交存储在对应的 stash 中；

(3)用户搜索对应子系统中的 stash，并读取目标数据块；

(4)用户对目标数据块进行读写操作，将操作后的数据块重新封装为新的数据块；

(5)用户为新数据块重新随机分配数据块和子系统的对应关系；

(6)用户将新数据块存入新子系统的 stash 中。

由于用户端存储新数据块的 stash 存储空间有限，因此每经过 S 次访问，用户须进行回收操作，将 stash 中的数据块写回对应的服务器的 ORAM 存储空间中，并将一次回收操作记为一个周期的结束。进行回收操作时，随机选取 $\sqrt{n}$ 个 stash，分别从中取一个有效数据块写回服务器。如果选取的 stash 中没有有效数据块，则需要写回一个无效数据块，以保证回收操作不会泄露信息访问方式。

4　ORAM 算法优化

在一些对 ORAM 性能要求比较高的领域，根据需求选择合适的 ORAM，并采用一系列优化策略对其进行专项优化，以适应应用场景。国内外研究人员在现有 ORAM 方案上，针对不同的优化目标提出了很多优化策略。

4.1　基于 Cuckoo Hash 的优化策略

CuckooHash(布谷鸟散列)是由 Pagh 和 Rodler[30] 在 2001 年为了有效缓解哈希冲突问题而提出的，能够有效提高计算效率和空间利用率。在 Hierarchical ORAM 中，为了避免哈希函数溢出[31]，每一层都是以 bucket 为操作存储单元，每个 bucket 中都有大量无效数据块，空间利用率低。搜索目标数据块时，需要先搜索目标数据块所在 bucket，再在 bucket 中遍历搜索，导致搜索效率不高。采用 CuckooHash 后，可以将数据块直接存储在每一层，有效减少了无效数据块数量。同时，查询时，可以直接通过 CuckooHash 查询目标数据块，搜索效率有所提高[32]。但是，直接存储的数据块特征明显[33]，不再具备 bucket 的特征一致性，使得安全性有所降低。

4.2　基于随机希尔排序的优化策略

在顺序存储结构的 ORAM 中，为保证数据块位置的安全性，每经过一个访问周期，都需要将数据块打乱进行重排操作，对访问效率有较大的影响。因此，复杂的重排操作是影响 ORAM 性能的主要因素之一。Goodrich[34] 在 2010 年提出的基于随机希尔排序的不经意排序算法，是一种与排序内容无关的高效排序方案，排序算法的复杂度为 $O(N\log N)$，而且常数项小于 10，提高了数据块重排的效率[35]。随着数据量不断增大，随机希尔排序算法的效率逐步降低，在大数据的情况下效率不高，同时容易出现不稳定的情况，影响访问。

4.3　基于逆字典序的优化策略

2013 年，Gentry[36] 针对驱逐操作提出逆字典序优化策略，每次进行驱逐操作的路径不再是随机的，而是根据模运算结果将二进制结果逆序表示后确定的路径。通过对驱逐操作的优化，可以尽可能地减少路径重叠，提高访问效率。通过算法确定驱逐路径，在一定程度上降低了操作的随机性，也就是牺牲了一部分的安全性。

4.4　基于多重迭代的优化策略

用户端主要进行对数据的更新操作，对用户端缓冲进行优化是提升性能的方向之一。用户端存储地址映射表，需要消耗大量空间资源，Devadas[37] 在 2016 年提出一种基于多重迭代的优化策略，将地址映射表的内容以小型 ORAM 存储在云服务器中。用户访问数据块时，先在存储地址映射关系的小型 ORAM 查询目标数据块的地址，然后在主 ORAM 中访问目标数据块。该策略通过多次迭代，可以降低用户端的地址映射表的空间开销，最低可以降低为 O(1)，并提高访问效率[38]。但是，多次迭代增加了数据交换次数，降低了访问效率，影响了用户体验。在大数据访问的情况下，不易被用户接受，一般只适用于密级较高的数据。

5　ORAM 方案的性能比较

ORAM 方案在理论上被证明是安全的。上文的方案均满足安全性要求，因此本节主要从存储空间和运算效率等方面对各方案开展性能比较。

综合现阶段主要 ORAM 算法的发展历程[20]，和用户对云存储数据的使用要求，本文归纳了 5 个指标，建立了 ORAM 算法性能评价指标体系，称为 SSIBT(写英文全称)。

(1)用户空间复杂度，反映用户存储和处理数据所需空间；

(2)服务器空间复杂度，反映服务器存储数据所需空间；

(3)交互复杂度，反映用户和服务器进行的数据通信轮数；

(4)带宽，反映每次访问用户和服务器传输的数据量；

(5)用户时间复杂度，反映用户处理数据的时间。

利用 SSIBT 指标体系，从所需时、空量级上对现阶段的主要算法进行分析和比较，结果如表 1 所列。

表 1 常见 ORAM 的性能分析比较

序号	ORAM	用户空间复杂度	服务器空间复杂度	交互复杂度	带宽	用户时间复杂度
1	Trivial ORAM	$O(1)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$
2	Square Root ORAM	$O(1)$	$O(N)$	$O(\log N)$	$O(\sqrt{N}\log N)$	$O(\sqrt{N}\log N)$
3	Hierarchical ORAM	$O(1)$	$O(N\log N)$	$O(\log N)$	$O(\log_3 N)$	$O(\log_3 N)$
5	Tree ORAM	$O(\log_2 N)$	$O(N\log N)$	$O(\log N)$	$O(\log_3 N)$	$O(\log_3 N)$
6	Path ORAM	$O(\log N)$	$O(N)$	$O(\log N)$	$O(\log N)$	$O(\log N)$
7	Ring ORAM	$O(\log N)$	$O(N)$	$O(\log N)$	$O(\log N)$	$O(\log N)$
8	Partition ORAM	$O(\sqrt{N})$	$O(N)$	$O(\log N)$	$O(\log_2 N)$	$O(\log_2 N)$

根据对各种典型 ORAM 方案的分析，数组存储结构的 ORAM 方案便于理解，操作方便，实现较为简单；但是，数据传输量较大，存储空间利用率不高，导致应用场景比较狭窄，一般适用于小规模数据的环境。

相较于数组存储结构，采用二叉树存储结构能够有效降低带宽，提高时间效率[39]。但是，用户增加了地址映射表，一定程度上牺牲了用户的存储空间和时间效率。同时，在二叉树存储结构中进行频繁的驱逐操作，复杂度很高，时间消耗大。通过对驱逐操作的优化，能提高效率，但不能从根本上解决问题。二叉树存储结构方案一般适用于高性能、对数据访问效率要求较高的应用环境[28]。

分区访问机制能够很好地契合云存储环境，进一步优化了带宽，减少了数据交互量。不同的区域采用不同的内置方案，对数据的保护更加严格。但是，Partition ORAM 在用户和服务器端需要大量的存储空间，而且每个区域的方案实现比较困难。该方案对服务器要求较高，一般适用于大数据存储访问环境。

6 未来研究展望

通过研究典型 ORAM 实现数据访问茫然性的方法，分析比较各种 ORAM 的性能，发现 ORAM 的性能还有很大的提升空间。现阶段的 ORAM 性能有了较大的提高，但是在一些对性能要求比较高的领域[40]，还需进一步提高。如何对现有 ORAM 进行性能优化，设计更为高效安全的、满足特殊应用需求的 ORAM 方案[41]，是目前 ORAM 研究的核心热点问题。

6.1 基于 k 叉树存储结构的 ORAM 方案

在二叉树存储结构的 ORAM 方案中，用户查询某一目标数据块时，服务器返回的数据块数量与层数相关，层数越多，数据交互量越大。在二叉树存储结构的 ORAM 方案中，为保护目标数据块的安全性，无论是否已经找到目标数据块，都需要从每一层返回数据库，直至叶子节点。通过将二叉树存储结构扩展为 k 叉树，可以把层数从原有的 $O(\log_2 N)$ 变成 $O(\log_k N)$，减少了数据交互量，缩短了用户的访问时间，提高了数据的访问效率。

6.2 具有强安全性的递归 ORAM 方案

针对一些极端重要的敏感数据，例如商业机密和军事数据，必须高度重视数据的安全性。通过牺牲一定的存储空间和访问时间，针对存储的不同数据类型，嵌套采用多种 ORAM 方案，多次读取数据，有效隐藏用户访问行为，极大地提高云端数据的安全性。

6.3 基于数据共享的多用户 ORAM 方案

现阶段，保护用户访问行为的 ORAM 主要是针对单用户，但在实际使用环境中，常常需要数据共享[42]。如何实现多用户 ORAM 方案[43]，具有很强的应用背景。为实现多用户访问[44]，需要引入代理加密机制，并借鉴权限分配思想[45]。设计基于角色的细粒度访问云端数据[46]的方案，不同用户访问数据的权力由数据所有者分配，在实现多用户共享云端存储[47]数据的同时，又能防止其他用户访问到数据所有者不想被他人知道的数据部分，从而保证了数据的安全性。

6.4 实用型 ORAM 安全计算系统

对安全计算系统的研究，主要是优化系统计算复杂度，提升系统效率，同时保证用户数据的隐私性。目前，ORAM 安全计算系统还处在理论验证和方案设计阶段[48]，还没有研究出结合实际应用的 ORAM 安全计算系统。可以将 ORAM 技术与现阶段成熟的基于秘密共享技术的多方计算系统 Sharemind[49]、多方计算系统 FairPlayMP[50] 等多方安全计算系统结合起来；也可以结合基于 MapReduce Spark 的安全计算系统模块进行改进，利用 ORAM 来保护读写访问行为。

6.5 高效 ORAM 数据结构的设计

在 ORAM 优化领域，除了对算法的优化，还有对数据结构的优化[51]，主要是通过结合实际应用情况，突破现有的结构，设计实现新的数据结构。早在 2014 年，Wang 等[52]就提出了基于位置的图结构理论，从图结构中划分簇，每个簇包含 $O(N)$ 个点，将簇作为访问的基本单元。为了提升增删数据块时的效率，Thang 等在 2017 年提出了基于表结构的数据库模式，其能够有效提高增删操作的效率。因此，如何结合实际应用情况设计高效数据结构，实现高效的 ORAM 方案，是研究的方向之一。

结束语 ORAM 在软件保护、安全计算、云存储安全等领域有着广泛的应用，是保护用户访问行为的最有效技术之一。但是，为了提高安全性，ORAM 增加的无效数据块、额外开辟的冗余存储空间、数据重加密操作、数据写回重组操作等，严重影响了 ORAM 的性能，限制了 ORAM 在实际场景中的应用。现阶段，ORAM 研究主要有两个方向：1）如何设计一种高效合理的 ORAM 结构，针对不同的应用场景选择合适的优化策略，在保证用户访问安全性的前提下，提高 ORAM 的性能；2）在一些特殊的应用场景下，应对强安全的应用需求，如何通过合理的访问结构和访问方式提升 ORAM 的安全性，确保数据具有强安全性。

参 考 文 献

[1] 刘书勇，付义伦. 基于 PKI 技术的可搜索云加密存储系统[J]. 软件导刊，2018，17(2)：182-185.

[2] 王斌，杨鹏，杨青. 基于密钥分离与加密策略的云存储加密方案[J]. 电信网技术，2015(9)：43-47.

[3] PASQUALE P，REFIK M，MELEK O，et al. CloudDedup：Secure Deduplication with Encrypted Data for Cloud Storage[P]. 2013.

[4] JUNG T，LI X Y，WAN Z，et al. Control cloud data access privi-

lege and anonymity with fully anonymous attribute-basedencryption[J]. IEEE Trans. on Information Forensics and Security，2015，10(1)：190-199.

[5] 刘赛，聂庆节，刘军，等. 基于量化行为的实时数据库备份系统访问控制模型[J]. 计算机与现代化，2018(1)：116-122.

[6] 李树凤. 抗访问模式泄露的 ORAM 技术研究[D]. 济南：山东大学，2016.

[7] GOLDREICH O，OSTROVSKY R. Software protection and simulation on oblivious RAMs [J]. Journal of the ACM (JACM)，1996，43(3)：431-473.

[8] 吴鹏飞，沈晴霓，秦嘉，等. 不经意随机访问机研究综述[J]. 软件学报，2018，29(9)：2753-2777.

[9] HUSSAIN S. A Low Performance-Overhead ORAM Design for Processor System with Un-trusted Off-chip Memory[C]// Proceedings of 2018 3rd International Conference on Computer Science and Information Engineering (ICCSIE2018). International Information and Engineering Association：Computer Science and Electronic Technology International Society，2018：12.

[10] 李红卫，古春生，景征骏，等. 云存储中基于 ORAM 的数据安全访问[J]. 微电子学与计算机，2014，31(6)：16-20.

[11] KUSHILEVITZ E，LU S，OSTROVSKY R. On the (in) security of hash-based oblivious RAM and a new balancing scheme [C]// Proc. of the23rd Annual ACM-SIAM Symp. on Discrete Algorithms. Society for Industrial and Applied Mathematics，2012：14-156.

[12] 宋宁宁. 基于全同态加密的 ORAM 方案[J]. 信息技术与网络安全，2018，37(11)：1-4.

[13] WANG X，CHAN H，SHI E. Circuit ORAM：On tightness of the goldreich-ostrovsky lower bound [C] // Proc. of the 22nd ACM Conf. on Computer and Communications Security. ACM Press，2015：850-861.

[14] GENTRY C，HALEVI S，JUTLA C，et al. Private database access with he-over-oram architecture[C]// Proc. of the 13th Int'l Conf. on Applied Cryptography and Network Security. Springer-Verlag，2015：172-191.

[15] 苑丹丹. 基于 ORAM 的隐私保护数据共享方案研究[D]. 济南：山东大学，2018.

[16] SHI E，CHAN T H，STEFANOV E，et al. Oblivious RAM with O((log N) 3) worst-casecost [M]// Advances in Cryptology-ASIA CRYPT 2011. Springer Berlin Heidelberg，2011：197-214.

[17] 宋衍. 基于属性的云存储访问控制与密文搜索研究[D]. 北京：北京交通大学，2018.

[18] 肖亮，李强达，刘金亮. 云存储安全技术研究进展综述[J]. 数据采集与处理，2016，31(3)：464-472.

[19] 刘全飞. 基于网络环境的计算机软件保护[J]. 信息与电脑(理论版)，2018(10)：173-174.

[20] 王倩倩. 茫然随机存取存储器加密方案的发展[D]. 烟台：烟台大学，2017.

[21] STEFANOV E，SHI E，SONG D. Towards practical oblivious RAM[EB/OL]. http://arxiv. orpjabs/1I06. 3652.

[22] SHI E，CHAN T H，STEFANOV E，et al. Oblivious RAM with O ((log N)3) worst-casecost[M]// Advances in Cryptology-ASIA CRYPT 2011. Springer Berlin Heidelberg，2011：197-214.

[23] DOERNER J. Scaling ORAM for secure computation [C]// Proc. of the 24th ACM Conf. on Computer and Communications Security. ACM Press，2017：523-535.

[24] ZHANG J，MA Q，ZHANG W，et al. TSKT-ORAM：A two-server kary tree ORAM for access pattern protection in cloud storage[C]// 2016 IEEE Military Communications Conference (MILCOM). IEEE，2016.

[25] TEEUWEN P：Evolution of oblivious RAM schemes[D]. Eindhoven：Eindhoven University of Technology，2015.

[26] STEFANOV E，VAN DIJK M，SHI E，et al. Path oram：An extremely simple obliviousram protocol[C]// Proceedings of the 2013 ACM SIGSAC conference on Computer & Communications Security. ACM，2013：299-310.

[27] LING R，FLETCHER C W，KWON A，et al. Constants count practical improverments to oblivious RAM[C]// Usenix Conference on Security Symposium. 2015.

[28] DAUTRICH J，STEFANOV E，SHI E. Burst ORAM：Minimizing ORAM response times for bursty access patterns[C]// 23rd USENIX Security Symposium (USENIX Security 14). 2014：749-764.

[29] MAAS M，LOVE E，STEFANOV E，et al. Phantom：Practical oblivious computation in a secure processor[C]// Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security. ACM，2013：311-324.

[30] PAGH R，RODLER F F. Cuckoo hashing[J]. Journal of Algorithms，2003，51(2).

[31] PINKAS B，REINMAN T. Oblivious ram revisited[C]// Proc. of the 30th Annual Cryptology Conf.. Berlin：Springer-Verlag，2010：502-519.

[32] KUSHILEVITZ E，LU S，OSTROVSKY R. On the (in) security of hash-based oblivious RAM and a new balancing scheme [C]// Proc. of the23rd Annual ACM-SIAM Symp. on Discrete Algorithms. Society for Industrial and Applied Mathematics，2012：143-156.

[33] GOODRICH M T，MITZENMACHER M. Privacy-Preserving access of outsourced data via oblivious RAM simulation[C]// Proc. of the 38th Int'l Colloquium on Automata，Languages，and Programming. Springer-Verlag，2011：576-587.

[34] GOODRICH M T. Randomized shellsort：A simple oblivious sorting algorithm[C]// Proc. of the 21st Annual ACM-SIAM Symp. On Discrete Algorithms. Society for Industrial and Applied Mathematics，2010：1262-1277.

[35] GOLDBERG I. Improving the robustness of private information retrieval[C]// Proc. of the 28th IEEE Symp. on Security and Privacy. IEEE，2007：131-148.

[36] GENTRY C，GOLDMAN K A，HALEVI S，et al. Optimizing ORAM and using it efficiently for secure computation[C]// Proc. of the 13th Int'l Symp. on Privacy Enhancing Technologies. Springer-Verlag，2013：1-18.

[37] DEVADAS S，DIJK M V，FLETCHER C W，et al. Onion ORAM：A constant bandwidth blowup oblivious RAM[C]// Proc. of the 13th Theory of Cryptography Conference. Springer-Verlag. 2016：145-174.

[38] REN L，FLETCHER C W，KWON A，et al. Constants count：Practical improvements to oblivious RAM[C]// Proc. of the 24th USENIX Conf. on Security Symp.. USENIX Association，2015：415-430.

[39] MOATAZ T，BLASS E O，MAYBERRY T. CHf-ORAM：A constant communication ORAM without homomorphic encryption[R]. 2015/1116，Cryptology ePrint Archive，2015.

（下转第 375 页）

结束语　单目标在监控区域中做匀速不定向运动，保证目标在整个运动路径中的任一时刻均能被 K 级覆盖到，且要求节点分布密度尽可能最小。本文提出的 KLDOA 算法依托于节点本身的特性，根据节点与目标之间的距离来实施节点旋转决策，从而使节点分布密度尽可能达到最小。通过仿真实验，可以验证该算法的有效性。在后期的研究工作中，我们会扩展目标个数，增加障碍物等更加现实的条件，以提升算法的通用性。

参 考 文 献

[1] LIU X. A Survey on Wireless Camera Sensor Networks[C]// International symposium on IT in medicine and education. Xining: Lecture Notes in Electrical Engineering, 2014: 1085-1094.

[2] 费娟，刘桂英，刘瑶. k 重覆盖设置算法的百分比覆盖研究[J]. 传感技术学报，2018，31(12)：1925-1930.

[3] 夏扬波，杨文忠，张振宇，等. 一种移动无线传感器网络的节点位置预测方法[J]. 计算机科学，2018，45(8)：113-118.

[4] CASTANO F, ROSSI A, SEVAUX M, et al. An Exact Approach to Extend Network Lifetime in a General Class of Wireless Sensor Networks[J]. Information Science, 2018: 433(4): 274-291.

[5] AI J, ABOUZEID A A. Coverage by directional sensors in randomly deployed wireless sensor network [J]. Journal of Combinatorial Optimization, 2006, 11(1): 21-41.

[6] CHEN U R, CHIOU B S, CHEN J M, et al. An Adjustable Target Coverage Method in Directional Sensor Networks[C]// IEEE Asia-pacific Services Computing Conference. Taiwan: IEEE Xplore, 2008: 174-180.

[7] HSU Y C, CHEN Y T, LIANG C K. Distributed Coverage-Enhancing Algorithms in Directional Sensor Networks with Rotatable Sensors[C]// International Conference on Distributed Computing and Networking. Hong Kong: Springer, 2012: 201-213.

[8] WU M C, LU W F. On target coverage problem of angle rotatable directional sensor networks[C]// Seventh International Conference on Innovative Mobile & Internet Services in Ubiquitous Computing. Taiwan: IEEE, 2013: 605-610.

[9] LIU L, MA H, ZHANG X. On Directional K-Coverage Analysis of Randomly Deployed Camera Sensor Networks[C]// IEEE International Conference on Communications. Beijing: IEEE, 2008: 2707-2711.

[10] FUSCO G, HIMANSHU G. Selection and orientation of directional sensors for coverage maximization[C]// IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks. Rome: IEEE, 2009: 1-9.

[11] FUSCO G, GUPTA H. Placement and Orientation of Rotating Directional Sensors[C]// IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks. Boston: IEEE, 2010: 1-9.

[12] WANG Z, BULUT E, SZYMANSKI B K. Distributed Target Tracking with Directional Binary Sensor Networks[C]// IEEE Global Telecommunications Conference. Honolulu: IEEE, 2009: 1-6.

[13] 蒋一波，陈琼，王万良，等. 视频传感器网络中基于移动目标轨迹预测 K 级覆盖增强算法[J]. 传感技术学报，2014(7)：956-963.

[14] 蒋一波，陈琼，王万良，等. 视频传感器网络中多路径 K 级覆盖动态优化算法[J]. 仪器仪表学报，2015(4)：830-840.

(上接第 347 页)

[40] WILLIAMS P, SION R, CARBUNAR B. Building castles out of mud: Practical access pattern privacy and correctness on untrusted storage[C]// Proc. of the 15th ACM Conf. on Computer and Communications Security. ACM Press, 2008: 139-148.

[41] WILLIAMS P, SION R. Access privacy and correctness on untrusted storage[J]. ACM Trans. on Information and System Security, 2013, 16(3): 12.

[42] 孙晓妮. 二叉树结构的多用户茫然 RAM 方案[D]. 济南：山东大学，2016.

[43] BOYLE E, CHUNG K M, PASS R. Oblivious parallel RAM and applications[C]// Proc. of the 13th Theory of Cryptography Conference. Springer-Verlag, 2016: 175-204.

[44] GOODRICH M T, MITZENMACHER M, OHRIMENKO O, et al. Privacy-Preserving group data access via stateless oblivious RAM simulation[C]// Proc. of the 23rd Annual ACM-SIAM Symp. on Discrete Algorithms. Society for Industrial and Applied Mathematics, 2012, 13(S1): 157-167.

[45] 孙晓妮，蒋瀚，徐秋亮. 基于二叉树存储的多用户 ORAM 方案[J]. 软件学报，2016，27(6)：1475-1486.

[46] BINDSCHAEDLER V, NAVEED M, PAN X, et al. Practicing oblivious access on cloud storage: The gap, the fallacy, and the new way forward[C]// Proc. of the 22nd ACM Conference on Computer and Communications Security. ACM Press, 2015: 837-849.

[47] SAHIN C, ZAKHARY V, ABBADI E, et al. Taostore: Overcoming asynchronicity in oblivious data storage[C]// Proc. of the 37th IEEE Symp. on Security and Privacy. IEEE, 2016: 198-217.

[48] 李红卫，上官经伦，古春生. 基于 ORAM 存储外包安全访问的研究[J]. 微电子学与计算机，2015，32(5)：6-10，15.

[49] BOGDANOV D, LAUR S, WILLEMSON J. Sharemind: A framework for fast privacy-preserving computations[C]// Proc. of the 13th European Symp. on Research in Computer Security. Springer-Verlag, 2008: 192-206.

[50] BEN-DAVID A, NISAN N, PINKAS B. FairplayMP: A system for secure multi-party computation[C]// Proc. of the 15th ACM Conf. on Computer and Communications Security. ACM Press, 2008: 257-266.

[51] 李红卫，叶飞跃，陈丹. 一种基于 ORAM 的数据可恢复性证明与访问模式的隐藏[J]. 电信科学，2013，29(12)：101-106.

[52] WANG X S, NAYAK K, LIU C, et al. Oblivious data structures[C]// Proc. of the 21st ACM Conf. on Computer and Communications Security. ACM Press, 2014: 215-226.