基于密钥状态散列树的密钥存储管理方案

王祥宇，马建峰，苗银宾，张凯，伍祈应
（西安电子科技大学网络与信息安全学院，陕西 西安 710071）

摘 要：针对密文检索系统中多源数据加密带来的海量密钥存储管理问题，提出了以密钥状态散列树作为密钥派生结构的密钥存储管理方案。该方案借助根密钥及密钥派生树进行密钥计算，且只需要存储根密钥和树结构，大大降低了密钥存储开销；另外，该方案可以根据撤销状态值进行密钥撤销管理，解决了派生树密钥撤销及结构更新难题。安全分析表明，部分数据密钥的泄露并不会泄露其他数据机密性，且基于实际数据集的性能分析表明所提密钥管理方案在实际应用中是可行的。

关键词：密钥存储管理；密文检索；密钥状态散列树；密钥撤销
中图分类号：TN918.4
文献标识码：A
doi: 10.11959/j.issn.1000-436x.2018080

Key storage management scheme based on keyed hash tree with state

WANG Xiangyu, MA Jianfeng, MIAO Yinbin, ZHANG Kai, WU Qiying
School of Cyber Engineering, Xidian University, Xi'an 710071, China

Abstract: To solve the problem of massive keys storage caused by multi-source data encryption in ciphertext retrieval system, a key storage scheme based on keyed hash tree with state was proposed. The scheme computes encryption key according to the root key and key derivation tree, and just needs to store the root key and the tree structure, which greatly reduces the key storage costs. In addition, the scheme manages key revocation according to the revocation state value, thereby solving the problem of key revocation and structure update. Strict security analysis shows that the partial data key disclosure does not leak the data confidentiality of remaining data, and the performance analysis using real-world dataset shows that the proposed key storage management scheme is acceptable in ciphertext retrieval system.

Key words: key storage management, ciphertext retrieval, keyed hash tree with state, key revocation

1 引言

随着云计算和大数据时代的到来，越来越多的云租户将数据外包到云服务器以减轻本地数据的存储和计算开销[1]。同时，为了解决数据机密性和可检索性之间的矛盾，密文检索技术[2,3]受到产业界和学术界的广泛关注。

由于数据的多源性（云数据来自多个数据拥有者），密文检索系统中不同的数据块通常采用不同的密钥进行加密以保障数据安全。然而海量云数据的加密存储会带来巨大的密钥存储管理问题。一方面，对海量数据进行加密会产生大量的密钥，这些密钥也会占用大量的存储空间，降低密钥存储开销对云密文检索系统是一个巨大挑战。另一方面，由于加密每个文件使用的密钥不一样，在使用过程中，解密者如何在大量密钥中快速调用相应的数据解密密钥也是一个重要问题。

目前，已经有了一些相对成熟的密钥管理系

收稿日期：2017-07-05；**修回日期**：2018-03-30
基金项目：国家高技术研究发展计划（“863”计划）基金资助项目（No.2015AA016007）；国家自然科学基金资助项目（No.61702404）；中央高校基本科研业务费基金资助项目（No.JB171504）；中国博士后科学基金资助项目（No.2017M613080）
Foundation Items: The National High Technology Research and Development Program of China (863 Program) (No.2015AA016007), The National Natural Science Foundation of China (No.61702404), The Central University Basic Business Expenses Special Funding for Scientific Research Projects (No.JB171504), China Postdoctoral Science Foundation Projects (No.2017M613080)

统。其中，单层派生关系（SDE, single-layer derivation equation）方案根据访问权限对用户进行分类并生成不同的密钥派生树，每种派生树对应一个用户掌管的密钥合集，通过密钥派生的方法来避免用户存储大量的加密密钥[4~6]。但是 SDE 方案只是在单授权者情况下减少了需要管理的密钥的数量，当授权者数量增加时，每个用户掌管的密钥也会增加，特别是当授权者数量很庞大时会产生海量密钥。建立在 SDE 方案基础上的双层派生关系（DDE, double-layer derivation equation）方案缩减了权限控制更新时的成本消耗[7,8]，但该方案的缺陷与 SDE 方案类似，都是在授权者数量很庞大的时候，无法有效减小用户掌管密钥的数量。

目前，在国内外安全研究领域中，许多方面都使用到了密钥派生的相关概念。除了上面提到的 SDE 与 DDE 方案中用到了密钥派生的思想，还有一些使用到了密钥树思想的研究方案[9~12]。

树是非线性的数据结构，由一个集合以及在该集合上定义的一种关系构成。集合中的元素称为树的节点，所定义的关系称为父子关系。父子关系在树的节点之间建立了一个层次结构。在这种层次结构中有一个节点具有特殊的地位，这个节点称为该树的根节点，或称为树根。树的这种结构特点能够将多个节点信息集中于某个节点上。如图 1 所示，通过将密钥映射到树的节点，建立密钥派生树，就能够实现将多个密钥信息集中于单个节点密钥上，大大降低密钥存储管理开销。

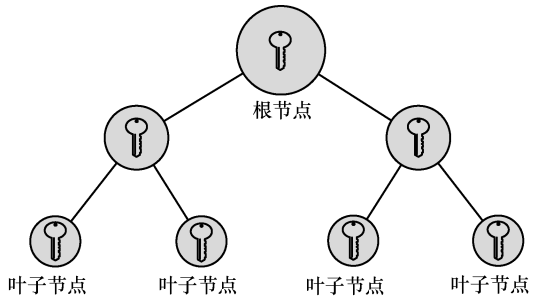


图 1　密钥派生树结构示意

文献[13,14]在基于状态密钥树的安全群组密钥分发协议研究中提出了状态密钥树的思想，该方法通过数学公式的计算来推导密钥派生树，减少了系统密钥的存储开销，但该方案会引入较大的计算开销。文献[15]针对高性能计算集群（HPC, high performance computing cluster）系统文件加密提出的 Horus 方案也使用密钥树进行密钥管理，大大减少了密钥的计算和存储开销。然而，随着密钥撤销次数的增加，此方案的密钥派生树的结构会向线性改变，密钥存储和管理开销会越来越大。

针对以上问题，本文提出了一种基于密钥状态散列树的密钥存储管理方案。该方案使用密钥状态散列树结构进行密钥派生，可以在不引入额外计算开销的前提下大大降低密钥存储开销。与此同时，与 Horus 方案相比，所提方案解决了由于密钥撤销带来密钥派生树结构改变而增大密钥存储管理开销的难题。表 1 显示了在密钥撤销情况下本文所提方案与 Horus 方案的密钥树情况对比，可以看到，本文所提方案的密钥树结构和数量不随密钥撤销而变化，因此，密钥存储开销不变，与 Horus 方案相比可以保证良好的性能。

表 1　　**密钥撤销情况下方案对比**

方案	密钥树结构	密钥树数量	密钥存储开销	根密钥/ bit
Horus	向线性变化	增多	增大	256
本文	不变	不变	不变	256

2　技术架构与安全威胁分析

本节分别介绍了所提密钥存储管理方案的技术架构以及安全威胁。

2.1　技术架构

本文密钥存储管理方案的技术架构如图 2 所示，系统分为 3 个部分：用户、私有云和公有云。

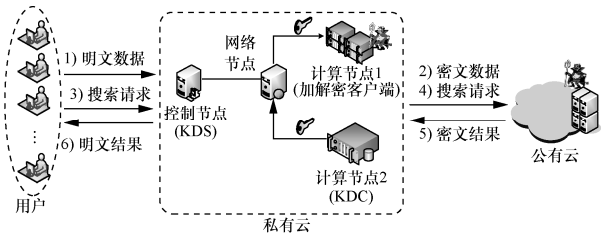


图 2　技术架构

系统流程如下所示。

1) 用户将自己的明文数据发送到私有云。

2) 私有云根据明文数据建立安全索引并加密数据上传到公有云进行存储。

3) 需要某一类数据时，用户向私有云发送明文搜索请求。

4) 私有云将其转换为搜索陷门后发送到公有云。

5) 公有云根据陷门和索引搜索到相应的密文结果返回给私有云。

6) 私有云对数据进行解密，将明文结果返回给用户。

在这个架构中，私有云用来分担用户在文件加密和搜索过程中的计算开销；公有云用来存储加密的数据并提供密文搜索服务，这符合一般的可搜索加密系统。私有云一般为高性能计算集群（HPC），以 OpenStack 为例，集群中包含控制节点、网络节点、存储节点和许多计算节点。

为了使云端无法获得除搜索模式外的任何与原始数据相关的数据，用户必须自己管理加密密钥。密钥分配过程在私有云中完成。方案采用了经典的客户端—服务器模型，客户端向密钥分配服务器（KDS, key distribution server）请求密钥，并使用 KDS 返回的密钥进行数据加密、解密。客户端运行在私有云当中的计算节点上，直接与用户进行交互；KDS 可以运行在私有云的控制节点或其他计算节点。当密钥需求大时，也可以单独地设置密钥分发中心（KDC, key distribution center）。本文中的 KDS 运行在控制节点上，客户端运行在计算节点上，KDC 由其他计算节点组成。

2.2　安全威胁假设

所提方案对安全威胁做出了以下假设。

1) 攻击者可能获得公有云存储的密文数据。恶意的云服务提供商可能会尝试解密用户的数据；外部攻击者可能攻破公有云服务器，获得其中所有数据的访问权。

2) 攻击者可能攻破某一解密计算节点。通常可搜索加密系统会假设私有云是完全可信的。但是，对安全威胁做一个更强的假设，假设攻击者（如恶意的用户）可以入侵私有云的计算集群并控制某一客户端，但是不能入侵所有客户端，假设 KDS 和网络节点是安全的。这个假设在现实中是常见的，因为 HPC 上运行的客户端程序是直接与用户进行交互的，给予用户很大的操作权限，恶意的用户可能利用客户端进行非法操作。而网络节点和控制节点掌管 HPC 的控制和通信，通常由高权限的管理员进行管理，普通用户没有控制权限，很难攻破。

3　密钥存储管理方案设计

图 3 展示了本文所提密钥存储管理系统的密钥分配过程。文件的加密和解密过程全部由运行在私有云上的客户端完成，分为文件加密上传和文件下载解密 2 个部分。

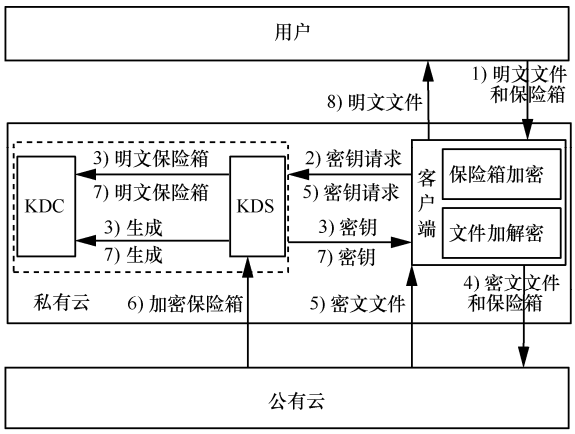


图 3　系统密钥分配过程

文件加密上传过程如下。

1) KDS 和用户分别公开自己的公钥 KU_{KDS} 和 KU_{A}，将私钥秘密保存。用户选择根密钥 K_{r}、密钥状态散列树的深度 d 和分支数 b 以及访问权限列表作为保险箱配置，将明文保险箱和文件发送给客户端。

2) 客户端将明文保险箱和密钥请求发送给 KDS。

3) KDS 根据配置生成密钥，返回给客户端，如果达到性能瓶颈，则使用空闲计算节点生成 KDC 进行密钥分发。

4) 客户端使用 KDS 返回的密钥加密对应文件，使用 2 个公钥 KU_{A} 和 KU_{KDS} 分别加密保险箱，将加密的保险箱和文件发送到公有云进行存储。

文件下载解密过程如下。

5) 客户端收到加密的文件后，向 KDS 请求解密密钥。

6) KDS 向公有云请求对应文件的加密保险箱，使用自己的私钥解密保险箱。

7) 根据保险箱中的权限列表判断客户端是否有得到密钥的权限。没有则终止，有则根据根密钥 K_{r} 以及派生树结构进行密钥派生，将密钥发送给客户端。如果达到性能瓶颈，则使用空闲计算节点生成 KDC 进行密钥分发。

8) 客户端根据接收到的密钥解密文件，发送给用户。

本文将详细介绍密钥存储管理过程中的密钥计算过程、根密钥的安全存储机制及密钥交换协议以及密钥分发集群 KDC 的设计。

3.1　密钥计算过程

本文设计的密钥存储管理系统采用带密钥的散

列树作为密钥派生结构，如图 4 所示。一个密钥树管理了大量的文件加密密钥，它们的加密密钥全部由根密钥派生而来，低层次的密钥控制更小范围的文件。叶子节点的的值就是对单个文件块的密钥。同一层次上的密钥控制的文件大小应该是相同的。为了尽量减少存储开销，每一层的分支数都是相同的。这样，不需要存储整个树形结构，只需要存储其层数和每个节点的叶子节点分支数即可推算整个树的结构。

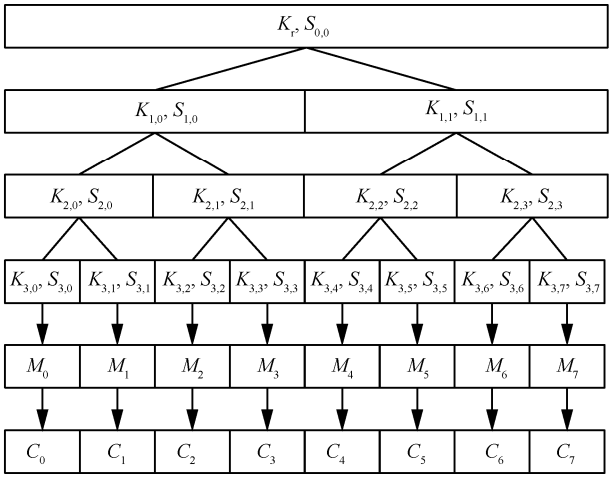


图 4　派生树结构

3.1.1　密钥派生过程

密钥派生树建立时，将密钥状态散列树的根节点设置为根密钥 K_r，所有节点的状态 $S_{x,y}$ 均为 0，表示从未撤销过。用父节点密钥值和状态值来计算出各子节点的密钥值，即计算带密钥的散列树中第 x 层中第 y 个节点对应的密钥值为

$$K(x,y)=H(K_{\text{parent}},x\|y\|*S_{x,y}) \tag{1}$$

其中，K_{parent} 是 $K(x,y)$ 的父节点对应的密钥，其为已知密钥，$H()$是 SHA-256 摘要算法，‖ 代表串联运算，这里使用 $x||y$ 是为了保证所有分块密钥互不相同，在 $S_{x,y}$ 前增加“*”是为了避免直接串联可能导致的 2 个密钥相同问题。

例如，图 4 中的 $K_{2,3}$ 由 $K_{1,1}$ 计算得出，即 $K_{2,3}=H(K_{1,1},2\|3\|*S_{2,3})$，以此类推，最终所有的分块密钥都可以由根密钥 K_r 计算得出。

3.1.2　密钥撤销过程

当用户对带密钥和状态的散列树中某些节点进行密钥撤销操作时，首先将对应节点的密钥撤销状态值 $S_{x,y}=S_{x,y}+1$，表示这个节点进行过一次撤销，并根据密钥计算过程中的公式重新计算密钥，作为文件加密密钥，分配给新用户，新用户用这个密钥加密自己的文件。而原有用户的密钥无法解密这些文件，保证了新用户文件的安全。

在实际系统中，状态值 S 的存储数据类型存在数据溢出，当密钥更新次数超过 $S_{x,y}$ 的数域时，在计算式中的 $S_{x,y}$ 前面多添加一个“*”，式(1)变为

$$K(x,y)=H(K_{\text{parent}},x\|y\|**S_{x,y}) \tag{2}$$

并且令对应的 $S_{x,y}=0$。再次超过数域时，重复此过程，保证系统可以一直运行。

3.2　根密钥的安全存储机制及密钥交换协议

本文使用保险箱（Lockboxes）保证了根密钥的安全性以及使用时的高效性[16]。根密钥是安全存储在公有云中的，私有云只承担密钥派生的计算开销，不需要承担存储开销。在这个过程中，涉及根密钥在公有云和私有云不同计算节点之间的分配。

为了满足分配中的需求，本文为密钥存储管理系统设计了密钥交换协议，协议符号含义如表 2 所示，密钥交换协议如图 5 所示。密钥交换分为文件加密上传以及文件下载解密这 2 个过程。

表 2　　密钥交换协议符号对照

符号	意义
K_r	根密钥
BS	密钥状态散列树配置包括深度和分支
K_{f,x_i,y_i}	文件群 f 对应的密钥
$<P_0,P_1\cdots>$	用户的访问权限列表
KU_A , KR_A	用户的公钥和私钥
$E_{KU_{KDS}}$, E_{KU_A}	使用对应的公钥加密
$D_{KR_{KDS}}$, D_{KR_A}	使用对应的私钥解密
KU_{KDS} , KR_{KDS}	KDS 的公钥和私钥

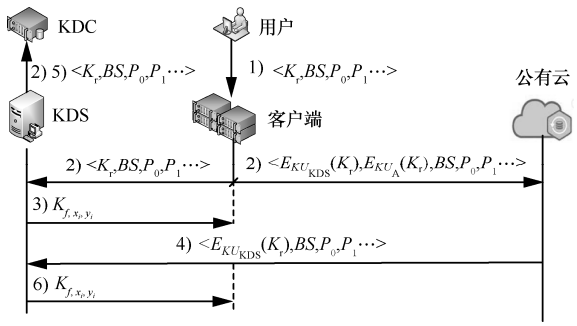


图 5　密钥交换协议

当一个新的文件群 f 被建立时，进行如下步骤。

1) 用户新建明文保险箱 $<K_r,BS,P_0,P_1\cdots>$ 作为文件群加密配置发送给客户端。

2) 客户端首先将明文保险箱发送给 KDS 并请求文件加密密钥，然后使用 KU_{A} 和 KU_{KDS} 分别加密根密钥，在完成加密之后将加密保险箱 $<E_{KU_{\mathrm{KDS}}}(K_{\mathrm{r}}),E_{KU_{\mathrm{A}}}(K_{\mathrm{r}}),\ BS,P_0,P_1\cdots>$ 发送给公有云进行存储。

3) KDS 返回 K_{f,x_i,y_i} 给客户端进行文件加密。需要注意的是，在步骤 2)中，如果 KDS 性能达到瓶颈，则把明文保险箱发送给新建的 KDC 增加密钥分发能力。

当一个私有云的客户端计算节点 C_i 受到用户的下载解密文件群 f 中的某一部分时，进行如下步骤。

4) KDS 从公有云得到加密的保险箱 $<E_{KU_{\mathrm{KDS}}}(K_{\mathrm{r}}),BS,P_0,P_1\cdots>$。

5) KDS 通过 $<P_0,P_1\cdots>$ 判断 C_i 是否拥有访问权限。如果客户端 C_i 被允许访问，KDS 使用自己的私钥解密保险箱中的根密钥 $<D_{KR_{\mathrm{KDS}}E_{KU_{\mathrm{KDS}}}(K_{\mathrm{r}})},\ BS,P_0,P_1\cdots>$。密钥 K_{f,x_i,y_i} 由 KDS 通过 K_{r} 和带密钥的散列树配置 BS 来计算。如果需要，将明文保险箱发送给新建的 KDC 进行密钥分发。

6) KDS 返回 K_{f,x_i,y_i} 给客户端进行文件解密。

这个协议的主要优势是 KDS 可以从本质上保持无状态，因此可以获得高并行性和安全性。需要注意的是，对于文件在密钥树中位置连续的情况，可以将权限内的密钥树中最高的节点密钥以及密钥树结构发送给客户端，客户端自行进行密钥派生，因为只进行一次通信，大大节省了通信开销，称之为快速方式。如图 4 中的密钥树结构，若客户端同时请求 $K_{3,6}$ 和 $K_{3,7}$，则可以直接将 $K_{2,3}$ 发送给客户端，由客户端自行计算分块密钥。

3.3 密钥分发集群 KDC 的设计

由于本文设计的密钥存储管理系统中，密钥管理功能与其他系统中数据管理功能是不相关的，所以密钥管理功能可以作为一组进程实现在一个或多个无状态的计算节点上，构成一个密钥分发中心。如果 KDC 到达了瓶颈，可以通过添加更多的节点来进行扩展。为了实现密钥派生，KDC 必须得到保险箱内容并确定用户访问权限。

在文件加密上传时，KDC 可以直接从客户端得到由文件拥有者提供的明文保险箱；在文件解密时，KDC 可以从 KDS 获得解密的保险箱。确定用户的访问权限的方法与之相似，可以通过保险箱中预先设定好的访问权限控制列表。

4 方案安全性与性能分析

本节对所提方案的安全性进行分析，并对方案使用真实数据集在真实环境下进行了详细的性能测试。

4.1 方案安全性分析

对于攻击者窃取公有云中数据的这种情况，在本文所提方案中数据不会明文存储在云端。因此，即使云服务器被攻破，也不能获得明文数据。只有通过文件的加密密钥解密文件，文件才是可读的。但是，要想得到加密密钥首先要得到根密钥，而要想得到根密钥必须首先解锁保存密钥的保险箱，保险箱只有通过合法用户的私钥才能打开[16]。

对于攻击者控制某些客户端计算节点这种情况。如前文所述，每个客户端只有它们拥有访问权限文件的密钥。由于散列函数的单向性，客户端无法得到这些密钥的父密钥，也就无法“扩展”相同级别的密钥从而得到权限范围之外的文件访问能力。这种方法在拥有大量节点的大规模高性能计算集群里十分有效，每个节点可能只需要访问整个文件群的 1%。如果系统中一个节点被攻破，入侵者就只获得了一小块数据[15]。

4.2 方案性能分析

本文所提密钥管理方案使用 C 语言进行开发；密钥交换协议利用 Socket 使用 UDP 来实现；使用 OpenSSL 中的 SHA-256 散列函数作为密钥派生的散列函数；为了满足细粒度的文件加密，使用 AES 的 XTS 模式来进行文件加密，源代码同样来自 OpenSSL。XTS-AES 是一种可调整分组密码，非常适合对那些被分割成固定长度（具体长度不限）的数据单元进行加密[17]。

为了贴合实际使用环境，在 OpenStack 集群中对本文所提方案进行了测试。采用 PC 机来模拟客户端，使用 KVM 虚拟机模拟私有云中的多个计算节点和公有云服务器。测试环境配置如表 3 所示，KVM 虚拟机只显示 CPU 的核数。

表 3 测试环境配置

使用者	操作系统	CPU	RAM/GB
用户	Ubuntu 16.04-64 位	Intel-core2-i5	8
KDS	Ubuntu 16.04-64 位	4 核	8
客户端	Ubuntu 16.04-64 位	4 核	8
公有云	Ubuntu 16.04-64 位	4 核	8

4.2.1　存储开销

存储开销分析如表 4 所示。这里的传统方案指的是逐个存储加密密钥的方案。从表 4 可以看出，相比于传统的存储方案，本文所提方案大大减小了密钥存储开销。

表 4　　存储开销分析

方案	分块密钥/bit	根密钥/bit	树结构/bit	综合开销/bit
传统方案	$256\times b^d$	0	0	$256\times b^d$
本文所提方案	0	256	$32+48\times\sum_{i=1}^{d}b^i$	$288+48\times\sum_{i=1}^{d}b^i$

本文以分支数为 2、4、6、8 的散列树为例，计算密钥状态散列树的存储开销占原始方案存储开销的百分比，结果如图 6 所示。

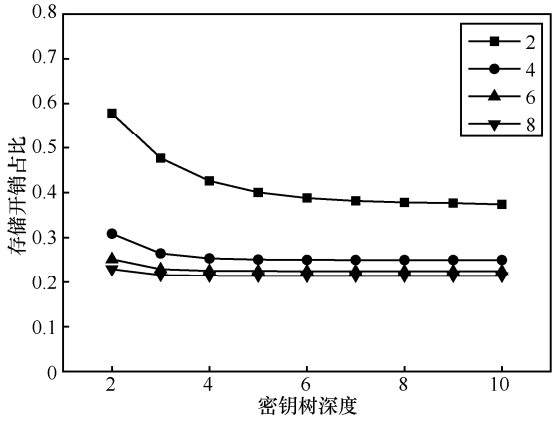


图 6　存储开销对比

从图 6 可以看出，随着带密钥和状态的散列树深度的增加，存储开销的减少越来越明显；对于相同的深度，分支数越多，存储开销越小，最小的大约为原始方案的 0.2，达到一定深度后，存储开销趋于稳定。另外，需要注意的是，带密钥和状态的散列树的根密钥由保险箱保护并存储在公有云之中。当需要使用时，由公有云发送给私有云进行密钥派生并解密文件，私有云不需要承担根密钥的存储开销，只承担树结构的存储开销和计算开销。

4.2.2　计算开销

本节对所提密钥存储管理方案的计算开销和性能进行分析，包括密钥派生 I/O、密钥分发 I/O 以及文件解密时间。首先，对密钥状态散列树的密钥派生 I/O 进行测试。

本文所提方案的密钥派生 I/O 如图 7 所示。从图 7 可以看出，方案的密钥派生 I/O 与派生树（密钥状态散列树）的深度有关，与派生树每层的分支个数无关。深度越深，I/O 越小，这是由于随着深度的增加，需要进行的散列运算次数也在增加。

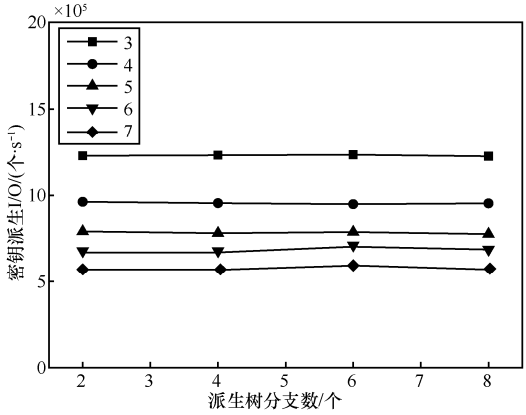


图 7　本文所提方案的密钥派生 I/O

为了更好地显示方案的性能，使用本文所提方案与 Horus 方案[15]进行对比。Horus 方案与本文所提方案类似，采用带密钥的散列树来进行密钥派生，但是，Horus 方案没有很好地解决密钥撤销问题。由于 2 种方案密钥派生效率与分支数无关，为了更好地对比 2 种方案的密钥派生效率，选取分支数 b=4 来进行对比测试，测试结果如图 8 所示。柱状图分别表示 Horus 方案和本文所提方案的密钥派生 I/O，折线图表示本文所提方案相比于 Horus 方案密钥派生 I/O 的减少量。从图 7 可以看出，本文所提方案比 Horus 方案的密钥派生 I/O 低 0.02~0.03，几乎没有影响。

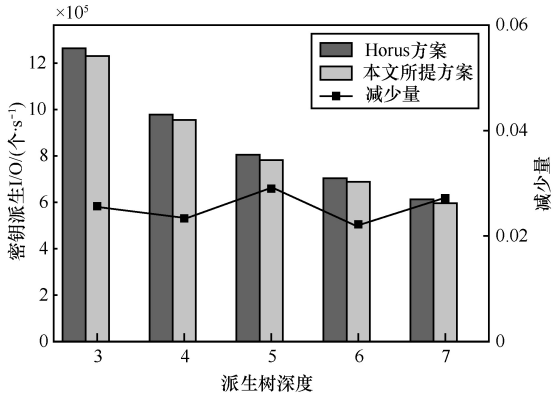


图 8　密钥派生 I/O 对比（分支=4）

图 9 显示了 2 种方案的密钥分发 I/O。从图 9 可以看出，Horus 方案和本文所提方案的密钥分发 I/O 几乎一致，这是由于在分发过程中需要将派生出的密钥发送到客户端，通信开销远大于密钥派生开销，因此密钥派生 I/O 的差距可以忽略不计。

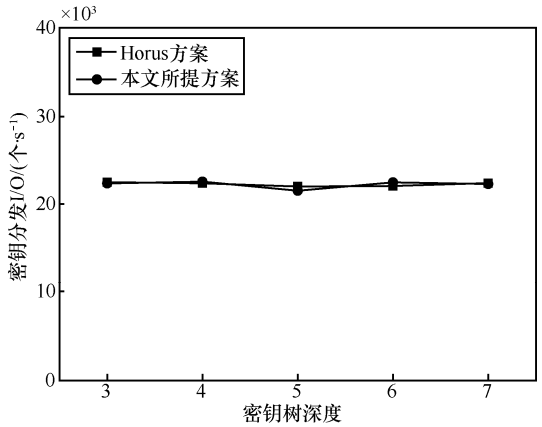


图 9 密钥分发 I/O 对比（分支=4）

图 10 显示了不同方案的解密时间开销分析（1 000 个文件，文件大小 2 KB），图中的 Horus 方案快速方案表示 Horus 方案的快速方式，即将高节点密钥传送给客户端进行解密的方式，本文快速方案表示本文所提方案的快速方式。从图 10 可以看出，通信开销占据了增加的解密时间开销的大部分，而密钥派生时间开销非常小。使用快速方案，由于只有一次通信的时间开销，可以达到与原始方案几乎相同的时间。但是，快速方案要求解密的文件是连续的，这在实际场景中很难达到。

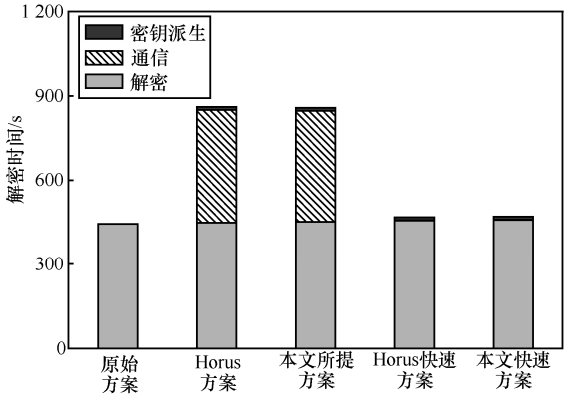


图 10 不同方案的解密时间开销分析

云环境中存储文件的复杂性很高，为了更好地反映本文所提密钥存储管理方案的性能，选取了不同大小的文件进行解密测试，文件大小从 1 KB 到 32 KB，文件数量从 100 到 100 000 个。考虑到实际场景中文件连续性难以预估，不考虑快速方案，测试结果如图 11 所示。图 11(a)~图 11(f)分别为文件大小为 1~32 KB 的测试结果。其中，柱状图表示原始方案、本文所提方案以及 Horus 方案的文件解密时间，折线图表示 2 种方案相比于原始方案的文件解密计算时间增量。这里的原始方案指的是将所有密钥全部存储在用户端的方案。从图 11 可以看出，随着文件的增大，本文所提方案和 Horus 方案相对于原始方案的文件解密时间增量逐渐降低，文件解密时间增量不受文件数量影响；2 种方案的解密时间基本相同，测试图中的波动来源于计算机性能的波动。

为了更全面地反映本文密钥存储管理方案的性能，选取从 1 KB 到 1 MB 大小不等的 1 000 个文件，使用原始的密钥管理方案以及本文所提方案和 Horus 方案分配密钥，分别进行了文件解密测试，测试结果如图 12 所示。

从图 12 可以看出，本文所提方案计算开销与 Horus 方案基本相同，本文所提方案由于在密钥派生过程中需要考虑撤销状态，计算时间略长。随着文件的增大，文件解密计算时间增加的比例逐渐减小，在文件大于 64 KB 之后，计算时间增量小于 0.2。可见，本文所提方案更适用于文件较大的存储系统。

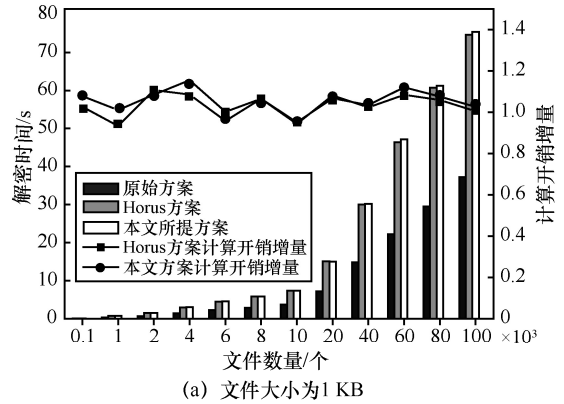
4.2.3 密钥撤销

Horus 方案采用新建密钥树的方式来解决密钥撤销问题，这将导致已有密钥树结构的改变以及密钥树数量的不断增长。本文所提方案很好地解决了密钥撤销问题。假设初始密钥派生树数量为 10 个，对每个派生树均进行不同程度的密钥撤销，图 13 显示了 2 种方案的密钥树数量对比。从图 13 可以看出，Horus 方案的密钥派生树数量随着每个密钥派生树撤销比例的增加，增长趋势呈现指数。在密钥撤销频繁的系统中，这将带来大量的密钥树管理开销以及密钥树存储开销。

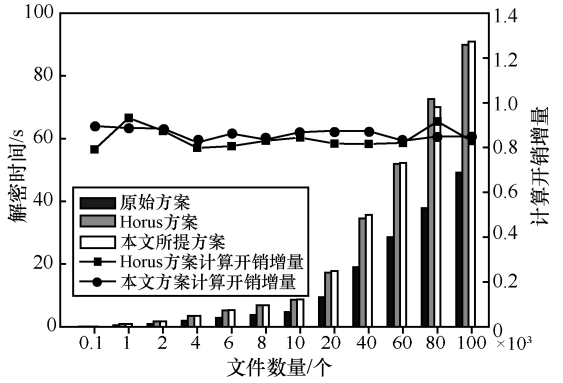
图 14 显示了不同撤销比例带来的密钥存储开销对比（以 7 层、4 分支的密钥树为例）。从图 14 可以看出，在撤销比例小于 0.3 时，Horus 方案的密钥存储开销略小于本文所提方案的密钥存储开销。当撤销比例大于 0.3 后，Horus 方案的密钥存储开销开始快速增长，在撤销比例达到 0.9 时，Horus 方案的存储开销是本文所提方案的 7 倍左右。

5 结束语

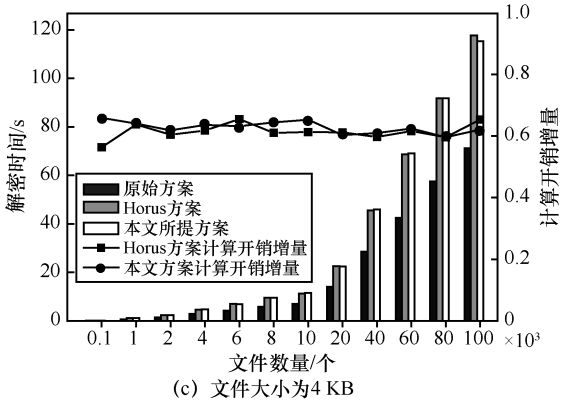
本文为解决海量多源数据加密带来的密钥存储管理问题，提出了一种基于密钥状态散列树的密钥存储管理系统。此方案使用密钥状态散列树作为密钥派生结构，只需要存储根密钥以及密钥树结构和撤销状态，分块加密密钥由这些参数派生计算得到，大大减少了密钥存储开销。同时，通过保险箱



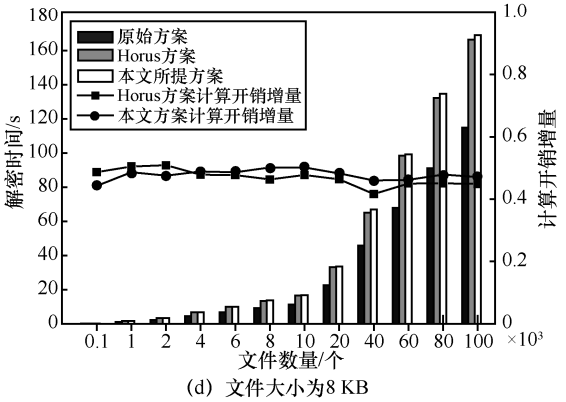
(a) 文件大小为1 KB



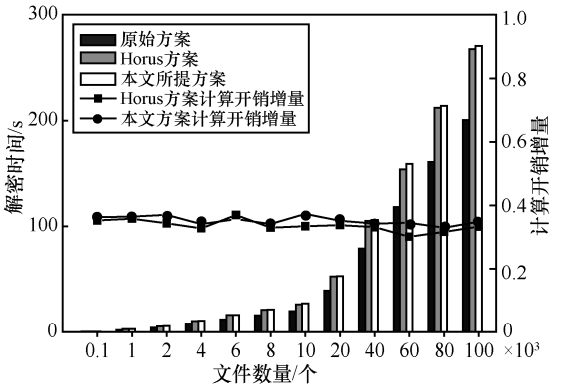
(b) 文件大小为2 KB



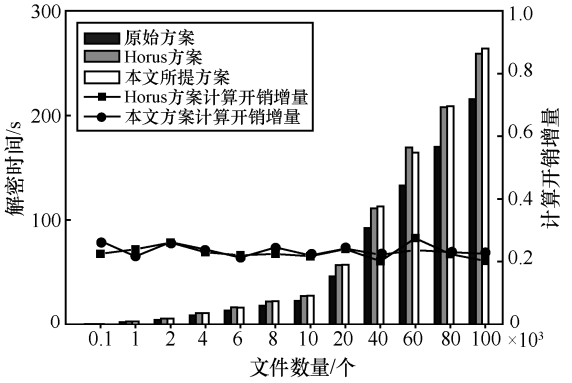
(c) 文件大小为4 KB



(d) 文件大小为8 KB



(e) 文件大小为16 KB



(f) 文件大小为32 KB

图 11　不同文件大小解密测试

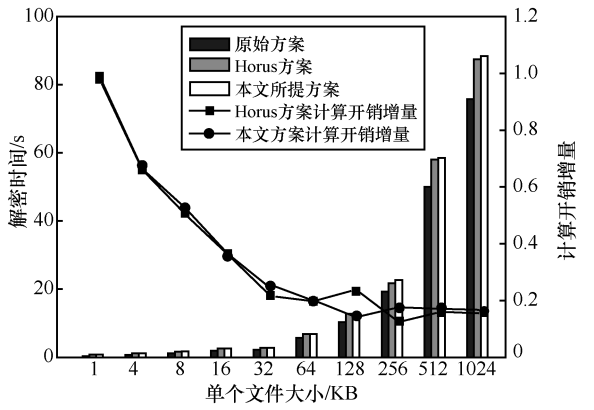


图 12　文件解密时间对比（1 000 个文件）

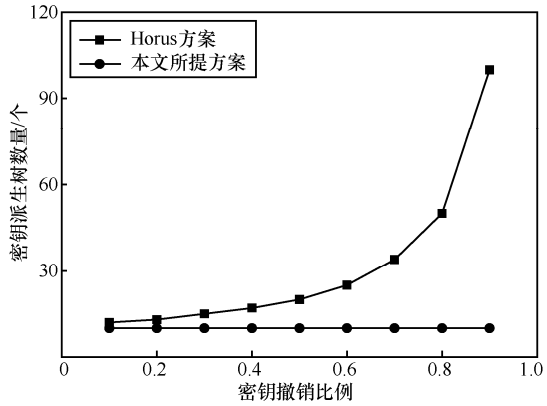


图 13　不同撤销比例密钥树数量对比

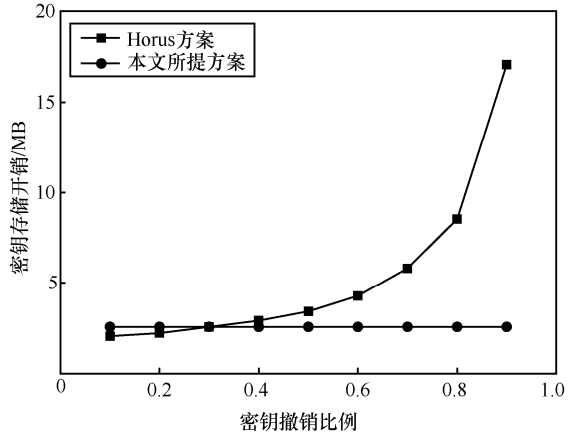


图 14　不同撤销比例密钥存储开销对比

以及密钥交换协议的设计，使密钥存储开销由公有云来承担，用户只需要承担很小的密钥计算开销。

经过详细的测试证明，在合适的密钥树结构下，本文所提方案可以减少将近 0.8 的密钥存储开销，并且这一开销由公有云承担，用户只承担较小的密钥计算开销。当解密的文件大于 64 KB 时，解密时间相较于逐一存储密钥的方案增加不超过 0.2。同时，本文所提方案解决了密钥撤销问题，与 Horus 方案相比，本文所提方案密钥树的结构与数量不随撤销比例而变化，在撤销比例大于 0.3 时，密钥存储开销优于前者。

参考文献：

[1] 曾文英, 赵跃龙, 尚敏. 云计算及云存储生态系统研究[J]. 计算机研究与发展, 2011, 48(S1):234-239.
ZENG W Y, ZHAO Y L, SHANG M. Research on cloud computing and cloud storage ecosystem[J]. Journal of Computer Research and Development, 2011, 48(S1):234-239.

[2] ABDALLA M, BELLARE M, CATALANO D, et al. Searchable encryption revisited: consistency properties, relation to anonymous ibe, and extensions[J]. Journal of Cryptology, 2008, 21(3):350-391.

[3] CURTMOLA R, GARAY J, KAMARA S, et al. Searchable symmetric encryption: improved definitions and efficient constructions[C]// ACM Conference on Computer and Communications Security (CCS). 2006:79-88.

[4] CUI Z, ZHU H, CHI L. Lightweight key management on sensitive data in the cloud[J]. Security & Communication Networks, 2013, 6(10):1290-1299.

[5] ATALLAH M J, BLANTON M, FAZio N, et al. Dynamic and efficient key management for access hierarchies[J]. ACM Transactions on Information and System Security (TISSEC), 2009, 12(3):1-43.

[6] BLUNDO C, CIMATO S, VIMERCATI S D C D, et al. Efficient key management for enforcing access control in outsourced scenarios[C]// Emerging Challenges for Security. 2009: 364-375.

[7] CUI Z, ZHU H, SHI J, et al. Lightweight management of authorization update on cloud data[C]// International Conference on Parallel and Distributed Systems (ICPADS). 2014:456-461.

[8] BESSANI A, CORREIA M, QUARESMA B, et al. DepSky: dependable and secure storage in a cloud-of-clouds[J]. ACM Transactions on Storage, 2011, 9(4):31-46.

[9] 马华, 白翠翠, 李宾,等. 支持属性撤销和解密外包的属性基加密方案[J]. 西安电子科技大学学报自然科学版, 2015, 42(6):6-10.
MA H, BAI C C, LI B, et al. Attribute-based encryption scheme supporting attribute revocation and decryption outsourcing[J]. Journal of Xidian University (Natural Science), 2015, 42(6):6-10.

[10] 付东来, 彭新光, 杨玉丽. 基于可信平台模块的外包数据安全访问方案[J]. 电子与信息学报, 2013, 35(7):1766-1773.
FU D L, PENG X G, YANG Y L. Trusted platform module-based scheme for secure access to outsourced data[J]. Journal of Electronics & Information Technology, 2013, 35(7):1766-1773.

[11] 田丰, 桂小林, 张学军,等. 基于兴趣点分布的外包空间数据隐私保护方法[J]. 计算机学报, 2014, 37(1):123-138.
TIAN F, GUI X L, ZHANG X J, et al. Privacy-preserving approach for outsourced spatial data based on POI distribution[J]. Chinese Journal of Computers, 2014, 37(1):123-138.

[12] 周林, 矫文成, 吴杨, 等. 一种基于层簇式的卫星网络组密钥管理方案[J]. 宇航学报, 2013, 34(4):559-567.
ZHOU L, JIAO W C, WU Y, et al. A group key agreement protocol based on layer-cluster for satellite network[J]. Journal of Astronautics, 2013, 34(4): 559-567.

[13] WONG C K, GOUDA M, LAM S S. Secure group communications using key graphs[J]. IEEE/ACM Transactions on Networking, 2000, 8(1): 16-30.

[14] 祝烈煌, 曹元大, 廖乐健. 基于状态密钥树的安全群组密钥分发协议[J]. 北京理工大学学报, 2006, 26(9):805-808.
ZHU L H, CAO Y D, LIAO L J. Secure group key distribution protocol based on the status key tree[J]. Transactions of Beijing institute of Technology, 2006, 26(9):805-808.

[15] LI Y, DHOTRE N S, OHARA Y, et al. Horus: fine-grained encryption-based security for large-scale storage[C]// Usenix Conference on File and Storage Technologies (FAST). 2013:147-160.

[16] KALLAHALLA M, RIEDEL E, SWAMINATHAN R, et al. Plutus: scalable secure file sharing on untrusted storage[C]//Usenix Conference on File and Storage Technologies. 2003:29-42.

[17] MARTIN L. XTS: a mode of AES for encrypting hard disks[J]. IEEE Security & Privacy, 2010, 8(3):68-69.

[作者简介]

王祥宇（1994-），男，内蒙古巴彦淖尔人，西安电子科技大学博士生，主要研究方向为数据安全、云安全、大数据隐私保护等。

马建峰（1963-），男，陕西西安人，博士，西安电子科技大学教授、博士生导师，主要研究方向为信息安全、密码学与无线网络安全等。

苗银宾（1988-），男，河南驻马店人，博士，西安电子科技大学讲师，主要研究方向为应用密码学、无线网络安全。

张凯（1987-），男，陕西西安人，西安电子科技大学博士生，主要研究方向为密码学、信息安全等。

伍祈应（1994-），女，湖南邵阳人，西安电子科技大学硕士生，主要研究方向为网络与信息安全。