可信固态硬盘:大数据安全的新基础

田洪亮 张 勇 许信辉 李 超 邢春晓

(清华大学计算机科学与技术系 北京 100084)

(清华大学信息技术研究院 北京 100084)

(清华信息科学与技术国家实验室(筹) 北京 100084)

摘 要 大数据平台,因其数据多、价值高和存储集中的特点,已经成为对攻击者非常有吸引力的目标.因此,大数据安全是一个非常重要的研究课题.然而,当前保障大数据平台(如 Hadoop)数据安全的两种常见方法各有不足:(1)访问控制.存在被外部黑客攻破或内部管理员绕过的风险;(2)数据加密.虽然安全性较高,但加密解密海量数据会增加显著开销.为了同时满足大数据应用对数据存储的高安全和高性能要求,文中提出可信固态硬盘(TrustedSSD),它提供安全增强的存储设备接口和协议,使得用户可以对存储中的数据施以细粒度的访问控制,从而保障存储中数据的安全.文中深入分析了可信固态硬盘的安全性,并详细介绍了系统设计与实现中的挑战和应对.实验结果表明,无论是在合成的还是真实的工作负载上,可信固态硬盘的运行开销不到 3%.因此,可信固态硬盘有望成为大数据安全的新基础.

关键词 大数据;数据安全;固态硬盘

中图法分类号 TP333 **DOI 号** 10.11897/SP.J.1016.2016.00154

TrustedSSD: New Foundation for Big Data Security

TIAN Hong-Liang ZHANG Yong XU Xin-Hui LI Chao XING Chun-Xiao

(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)

(*Research Institute of Information Technology, Tsinghua University, Beijing* 100084)

(*Tsinghua National Laboratory for Information Science and Technology (TNList), Beijing* 100084)

Abstract Big data, known for its characteristics of high volume, high value and centralized storage, is an attractive target for attackers. Thus, big data security is an important issue. However, the two most common data security measures used in big data platforms (e.g. Hadoop) are not satisfactory: (1) Access control mechanisms are prone to bugs and vulnerabilities; (2) Data encryption is provably secure but incurs considerable overheads during data processing. In this paper, we present TrustedSSD, a secure Solid State Drive (SSD) that efficiently guarantees the security of data-at-rest by enforcing a fine-grained access control. We first analysis the security of TrustedSSD, and then describe the design and implementation of the system, highlighting the challenges involved. We built a prototype of TrustedSSD on a commercially successful SSD controller. Experimental results on both synthetic and real-world workloads show that TrustedSSD incurs less than 3% overhead. Therefore, we believe TrustedSSD is a promising approach to big data security.

Keywords big data; data security; solid-state drive (SSD)

收稿日期:2015-01-12;在线出版日期:2015-07-13.本课题得到国家"九七三"重点基础研究发展规划项目基金(2011CB302302)、国家"八六三"高技术研究发展计划项目基金(SS2015AA020102)和清华大学自主科研计划资助.**田洪亮**,男,1987 年生,博士研究生,主要研究方向为云数据安全、基于新硬件的数据处理.E-mail: tatetian@gmail.com.**张 勇**,男,1973 年生,博士,副研究员,主要研究方向为数据管理和数据分析.**许信辉**,男,1990 年生,硕士研究生,主要研究方向为数据管理.**李 超**,女,1978 年生,博士,副教授,主要研究兴趣包括存储系统和数据管理.**邢春晓**,男,1967 年生,博士,研究员,博士生导师,主要研究领域为数字图书馆和数据库技术.

1 引　言

大数据时代已经来临.为了将快速增长的信息转换为价值,企业和组织往往将各种类型的数据,包括个人、财务、医疗等敏感信息,整合到统一的大数据系统.这种集中存储的大量敏感信息,无疑将成为攻击者的理想目标.因此,保护大数据的安全性,尤其是数据机密性,是一个重要的研究课题.

在典型的大数据平台(比如 Apache Hadoop)中,海量数据是存储在廉价服务器集群中各个节点的本地硬盘中的.为了保护存储介质中的数据,防止敏感数据泄露,目前主要使用两种手段:访问控制和数据加密.但这两种方法都各有不足.访问控制,通常由系统软件实施,其安全性依赖于代码的安全性和健壮性.随着系统越来越庞大复杂,很难保证不存在软件错误或安全缺陷,即使像 Google 这样技术先进的公司也不能幸免①.除了软件可能存在问题外,企业内部的管理员也可能利用特权绕开访问控制,窃取隐私数据,这从瑞士银行员工泄露用户资料事件可见一斑②.为了弥补系统软件访问控制的不足,一个常见做法是数据加密.基于密码学的方法虽然具有更高的安全性,但对海量数据加密解密不可避免地带来了显著的额外开销.以 Hadoop 为例,其在设计安全机制时的一大考虑因素是对性能的影响(不超过 7%)[1].到目前为止最新的 2.4 版本,Hadoop 仍然没有提供默认的加密解密框架,部分原因是顾忌加密解密的开销.综上所述,现有方法难以在保护海量数据时既提供较高的机密性保证,又只产生可忽略不计的额外开销.

通常来讲,服务器的本地存储介质(通常是硬盘)可以被主机任意读写,因此是不可信的,需要系统软件的访问控制或数据加密对其保护;但理想的情况是,硬盘是应该且可以增加安全机制的.近些年,固态硬盘,因其延迟低、吞吐量大、能耗低、噪音小,在各种应用场景正逐渐取代传统的机械硬盘,包括企业级服务器市场③.而且,最近的一些针对固态硬盘的存储内计算(In-Storage Processing)的一系列研究[2-5]表明,固态硬盘内有充足的资源(如 CPU 和内存等)支持额外的功能.

基于以上观察,本文提出一种解决大数据平台数据存储安全的新思路——存储内安全(In-Storage Security)把对数据的访问控制从主机上的系统软件下放到底层存储内部.更具体地说,我们设计并实现了可信固态硬盘(TrustedSSD),它提供安全增强的存储设备接口和协议,使得用户可以对存储中的数据施以细粒度的访问控制,保护存储中数据的安全性.访问控制层(Access Control Layer,ACL)是可信固态硬盘的安全引擎,与普通固态硬盘具有存储引擎,即闪存转换层(Flash Translation Layer,FTL),紧密结合在一起,实现高效的用户认证和操作授权.结合用户库函数和操作系统支持,可信固态硬盘可以为数据密集型应用(比如 Apache Hadoop)提供安全的数据存储服务.

本研究的主要贡献有:

(1)我们为可信固态硬盘设计了安全增强的存储设备接口和协议,详细分析和论证了其安全性,展望了其在大数据平台中的应用前景(见第 3 节"存储内安全").

(2)我们在可信固态硬盘的设计与实现过程中解决了数项技术难点,包括提高硬件平台的基准性能,设计高效的访问控制机制,修改复杂的操作系统 I/O 栈以及扩展硬盘接口(见第 4 节"设计与实现").

(3)我们在商用成功的固态硬盘控制器上实现了可信固态硬盘的原型系统.在合成的和实际的工作负载以及 HDFS 上测试了可信固态硬盘的性能,实验结果表明可信固态硬盘的安全机制带来的运行开销小于 3%(见第 5 节"实验结果").

传统的观点认为,存储是不可信的,大数据的安全保障应该由系统软件负责.但我们在可信固态硬盘上的工作表明,存储不仅可以信任,而且可以有效地保障数据安全,成为大数据安全的新基础.

2 相关工作

大数据系统的研究主要有 3 个方向:存储、管理和分析[6].因而,其安全和隐私保护也可以从这 3 个方面着手.数据分析层面的隐私保护技术,如匿名化[7]和差分隐私(Differential Privacy)[8]等,以及数据管理层面的安全保护技术,如加密数据查询[9]和

① Google blames software update for lost gmail data. http://www.cnet.com/news/google-blames-software-update-for-lost-gmail-data/, 2011-02-28

② Germany Tackles Tax Evasion. http://online.wsj.com/news/articles/SB10001424052748704197104575051480386-248538, 2010-02-07

③ Flash Pricing Trends Disrupt Storage. http://wikibon.org/wiki/v/Flash_Pricing_Trends_Disrupt_Storage

可信硬件[10]等,可以加强大数据系统的安全性和隐私性,但无法完全替代在存储层面的安全措施的作用.业界对系统安全的共识是,没有一种安全措施能够防范所有安全威胁,多层次的安全机制才能最大限度地保证系统安全.可信固态硬盘位于大数据系统最底层,即存储层.本节将简述存储层面的各种安全技术,并与可信固态硬盘做比较.

目前,业界针对硬盘本身的安全性增强技术主要有两种:全盘加密(Full-Disk Encryption)①和作为硬盘接口标准 ATA 一部分的安全特性集[11].全盘加密是一种对硬盘上所有数据的、对用户透明的加密技术.全盘加密的硬盘在启动时会要求用户输入密码,密码匹配后,后续的读/写操作都以用户密码对应的密钥解密/加密硬盘中的数据.这样,若硬盘本身不慎丢失或被恶意窃取,硬盘中的数据由于已加密就不会泄露.除了在存储中做数据加密,存储接口标准也可以增强安全性.广泛使用的存储接口标准 ATA 有可选的安全特性集,能够锁住存储设备(拒绝任何读写请求),直到用户用密码解除锁定.前述的全盘加密或 ATA 安全特性集都有助于提升硬盘的安全性,它仍然有明显不足.这是因为,这两种硬盘保护方法在用户输入完密码之后就在安全性方面与普通硬盘无异了,入侵者或恶意程序可以在用户输入密码之后(通常就是开机之后),窃取、篡改或删除硬盘中数据.

此前,有许多针对传统硬盘的增强安全性的研究工作,但它们在目的、方法和适用范围上都与本研究不同.自安全存储(Self-Securing Storage)[12]在一段时间内保留数据的历史版本,以应对入侵者在未发觉的情况下篡改或删除数据.网络安全硬盘(Network-Attached Secure Disks)[13]通过把访问控制下放到各个硬盘,从而消除了在传统网络文件系统中访问控制完全集中在文件服务器这一系统瓶颈.可存活存储系统(Survivable Storage System)[14]把一个文件的每个片段打散分布到多个存储节点,这样即使单个存储节点被入侵,也可以保证数据的隐私性、可靠性和可用性.

前面提到的硬盘方面的研究都假设存储设备的接口是基于块的,基于块的接口最早可以追溯到 20 世纪 50 年代,而 2000 年左右基于对象的存储设备接口开始得到产业界和学术界的兴趣,继而产生了一种新的存储类型,即对象存储(Object Storage)[15].就本文所关注的安全方面而言,对象存储的最大优点是以对象为单元的访问控制.虽然对象存储的思想被广泛接受,比如 Google File System 中文件元数据和文件数据块的分开存储的架构[16],再如 Amazon Web Service S3② 的基于对象的存储接口,但在硬盘层面基于对象的接口仅停留在标准化阶段[17],没有大规模产品化,而基于块的接口仍然是硬盘设备的绝对主流.可信固态硬盘在保持块接口的前提下实现了数据的细粒度访问控制.无论是分布式文件系统 Google File System,还是云存储系统 Amazon Web Service S3,这些借鉴了对象存储思想的大数据系统均可以通过采用更安全的底层存储设备,如可信固态硬盘,来提高系统整体的安全性.

最近几年,有不少研究工作探索如何为固态硬盘增加额外的功能.文献[2]在固态硬盘内增加硬件逻辑以更高效地处理数据扫描操作.文献[3]研究把关系型数据库的部分运算下推到固态硬盘内部完成.文献[4]扩展了 Hadoop,使得部分 Map 任务可以在固态硬盘内部完成.文献[5]提出了事务型闪存转换层(Transaction FTL),把 SQLite 保证事务原子性(Transaction Atomicity)的功能放到了固态硬盘中.与上述针对固态盘的存储内处理(In-Storage Processing)的研究工作不同的是,可信固态盘是一种存储内安全(In-Storage Security)的尝试.

伴随着闪存技术的不断成熟,固态硬盘得到了市场越来越广泛的接受,与其有关的安全方面的问题也引起了学术界的兴趣.文献[18]研究了固态硬盘特有的闪存转换层(Flash Translation Layer)对实现可靠擦除(Reliable Erasing)的挑战.文献[19]探讨了闪存单元的允许操作次数有限而带来的安全风险(如拒绝服务攻击).与上述研究不同的是,本研究的焦点是固态硬盘层面的访问控制.从思路上看,与本研究最接近的是 Butler 等人[20-21]的工作,其中也涉及了基于闪存的硬盘访问控制机制,但 Bulter 等人的工作仅停留在框架性描述的层面,缺乏接口协议的完整分析、系统实现的详细讨论,和实验数据的有效支撑.而这些正是本研究工作的主要贡献.

3 存储内安全

存储内安全(In-storage Security)是指一种在

① Disk encryption. https://en.wikipedia.org/wiki/Disk_encryption

② Amazon Simple Storage Service(S3). http://aws.amazon.com/s3/

存储设备内部实现安全机制的做法. 安全界的一项共识是，没有一种安全措施能够防范所有安全威胁，多层次的安全机制才能最大限度地保证系统安全. 传统上，普通存储介质（如硬盘）被认为是不可信任的；而本文认为，存储介质的不可信正是大数据安全的缺失环节. 存储内安全可以有效地增强存储中数据的安全性.

本文主要关注于利用可信固态硬盘保护存储中数据的机密性. 本节首先介绍可信固态硬盘应对的安全威胁，然后讨论如何设计可信固态硬盘的接口和协议以有效地保证数据机密性，最后以 Hadoop 为例说明可信固态硬盘在大数据平台中的应用.

3.1　安全威胁

假设攻击者：

(1) 可以获得主机的特权. 他们既可能是恶意的外部入侵者，也可能是好奇的内部管理员. 他们可能在服务器上运行恶意程序直接读取数据文件，也可能修改操作系统或数据库管理系统的权限. 无论手段如何，结果都是他们可以成功地绕开所有系统软件对存储中数据的访问控制.

(2) 企图获取存储中的敏感数据，但对篡改或销毁数据并不感兴趣. 作者认为这个假设是比较符合实际的，因为读取数据很难被检测到，而篡改或删除数据相对容易被发现，作案风险大、利益小.

(3) 只能通过定义良好接口访问存储中的数据，即假设攻击者不能物理攻击（比如盗窃或拆解）. 可信固态硬盘主要是用作集群中数据节点上的直连存储设备，通过常见的存储接口（SCSI、SATA 和 SAS 等）与主机连接. 这个定义良好的存储接口是存储与主机交互的唯一途径，也是对存储设备的唯一攻击面（Attack Surface）.

(4) 攻击者无法篡改可信固态硬盘的固件程序（Firmware），因为固件程序是受到保护的. 一个比较理想的保护方式是，可信固态硬盘的新固件必须在能被验证是合法的（比如有生产厂家数字签名）之后才能替换原来的固件. 另一个可能的方式是，由可信固态硬盘上的一个物理开关决定固件是否处于可更新的状态. 可信固态硬盘的原型系统采用了第 2 种方式. 由于攻击者不能物理接触可信固态硬盘，也就无法篡改固件.

3.2　安全协议

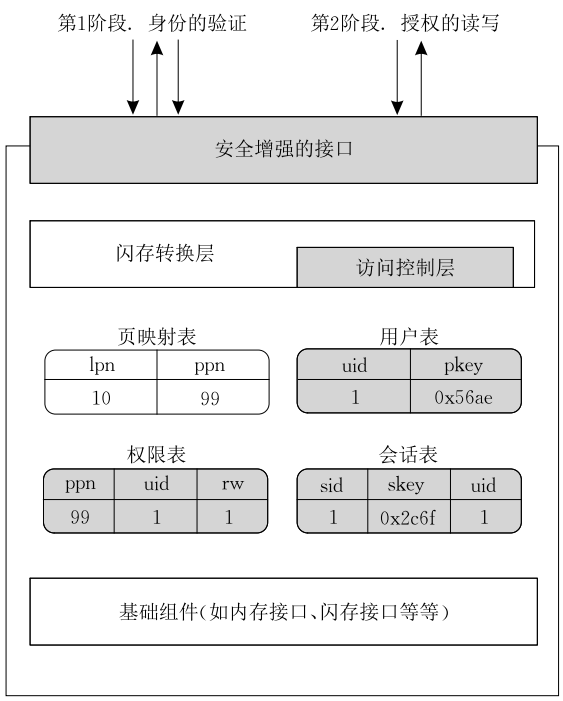
本小节讨论如何设计一个安全增强的存储接口和协议，使得在不可信的主机上对存储安全地访问成为可能. 可信固态硬盘的接口是主机与它交互的唯一途径；主机必须按照这个接口的协议、通过发送合法命令，才能读取或改变存储内部的状态. 我们希望扩展普通的存储接口，使其具备用户验证、操作授权等功能，成为一个安全增强的存储接口.

然而，设计安全增强的存储接口和协议是有多重挑战的. 首先，攻击者可以监听主机和存储之间的所有交互；重放攻击（Replay attack）、伪装攻击（Impersonation attack）和中间人攻击（Man-in-the-middle attack）等各种攻击都是可能的. 其次，存储器每秒钟处理的 I/O 操作数可以高达 10 000，而其内部的计算资源有限，因此协议对每个 I/O 操作的验证授权开销必须很小. 最后，安全增强的存储协议必须与原有的软硬件接口尽量兼容，使得驱动程序的改动尽可能的小，并且不需要任何硬件上的改动.

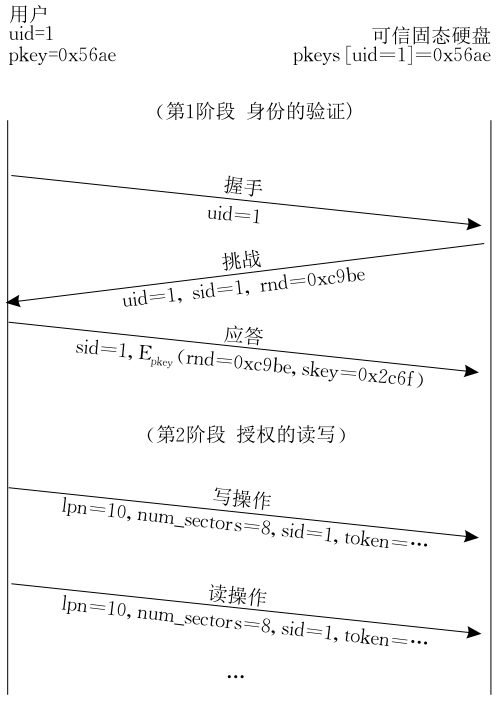
最终设计出来的可信固态硬盘的安全增强的接口和协议的概况如图 1 所示. 用户对存储的访问分两个阶段：

第 1 阶段，身份验证. 这个阶段的目的是认证用户身份，并建立会话. 每个用户都有一个 ID（uid）和密钥（pkey），在用户向可信固态硬盘注册账号的时候完成分配；用户 ID 是公开的，但用户密钥只由用户和可信固态硬盘（图 1(a)中的用户表）掌握. 用户的身份验证采用密码学的经典方法——挑战-应答协议（如图 1(b)）. 挑战-应答协议，简单地说，就是通过让用户使用他的密钥加密一个随机数来证明它拥有这个密钥的事实，从而让服务方可以认证用户的身份. 完成上述身份验证之后，用户和可信固态硬盘之间建立起了一个会话，用户和可信固态硬盘（图 1(a)中的会话表）都拥有这个会话的密钥.

第 2 阶段，授权读写. 用户凭借会话密钥进行读写操作，可信固态硬盘只授权合法的读写操作. 普通的块设备的读写操作，逻辑上可以简化地看成调用 read(lpn，num_sectors) 和 write(lpn，num_sectors，data) 两个函数，其中 lpn 是逻辑页地址（Logical Page Number），num_sectors 是读写的扇区数目，一个逻辑页包含数个扇区（比如 8 个），一个扇区的大小是 512 字节. 为了验证读写操作的合法性，可信固态硬盘为 read 和 write 扩充了两个可选参数（如图 1(b)），会话 ID（sid）和操作令牌（token），前者是



(a) 可信固态盘的架构图



(b) 用户交互示例

图 1 可信固态硬盘的安全机制(图 1(a)中方角和圆角方框分别表示模块和元数据;灰色的方框表示与安全相关;圆角方框中的数值表示图 1(b)所示的交互完成时的状态. 图 1(a)中的 lpn、ppn、uid、pkey、rw、sid 和 skey 分别表示逻辑页地址、物理页地址、用户 ID、用户密钥、读写权限、会话 ID 和会话密钥. 图 1(b)中(不在图 1(a)中)的 lpn、pkeys、num_sectors、rnd、E_{pkey} 和 token 分别表示起始逻辑页地址、用户表、读/写扇区数、随机数、对称加密算法和操作令牌)

公开的,后者是根据会话密钥生成(生成规则见下一小节"安全分析"). 可信固态硬盘在接收到读写命令后,根据会话 ID 和操作令牌检查操作是否属于一个合法的会话,如果不合法,则拒绝执行;如果合法,则用会话表(见图 1(a))查出发出这个操作的用户 ID. 对于合法的写操作,页权限表中把涉及到的页标记成属于这个用户;对于合法的读操作,则检查涉及到的页是否全都属于这个用户,如果是,则继续执行这个读操作;否则,拒绝执行.

如上所述的根据操作用户是否为扇区拥有者来决定扇区是否可以访问是最基本的访问控制策略. 为了允许数据更大程度的共享,还有多种方法可以扩展上述访问控制策略. 比如,维护扇区的读写权限(图 1(a)中权限表的 rw 字段,长度是 2 个 bit):当 rw=00 时,表示只有扇区拥有者可以读写;当 rw=10 时,表示其他用户也可读,但不可写;当 rw=01 时,表示其他用户可以写,但不可读;而当 rw=11 时,表示所有用户都可以读写,相当于没有任何访问控制. 除了读写权限外,增加用户组的概念也是一种可能性,类似于 UNIX 文件系统的访问控制策略,这里不再赘述.

可信固态硬盘的访问控制完全与普通硬盘的使用模式兼容. 当接收到普通读写命令,其会话 ID 和操作令牌等参数都是空,可信固态硬盘会认为这种读写命令来自公共用户(uid=0). 权限表的用户 ID 字段 uid 的初始值都是 0,读写权限字段 rw 为 11,也就是说所有扇区默认属于公共用户,且可以被任意读写. 私有用户(uid≠0)可以任意读写公共用户的扇区,而私有用户的扇区受到访问控制机制的保护.

3.3 安全分析

采用基于会话的安全机制是出于两方面的考虑. 一方面,大数据应用的数据处理模式往往是把计算任务发送到数据源(在 Apache Hadoop 中是 Map/Reduce 任务发到数据节点). 计算任务为了合法地读取可信固态硬盘中的数据必然需要某种凭证(在不可信主机上),如果凭证是用户密钥,则一旦丢失,危害很大;如果凭证是会话密钥,即使被窃取,攻击者也仅能在相对短暂的会话期间内冒充合法用户. 另一方面,密钥越长安全性越好,但各种密码学操作的开销越大. 把验证身份的密钥(用户密钥,使用频率低)和授权读写操作的密钥(会话密钥,使用频率高)区分开,前者使用长位数(比如 128 个比特),后者使用短位数(比如 32 个比特),可以在保证

安全性的前提下，把开销降到最低. 为了防止短位数的会话密钥被暴力破解，可信固态硬盘一旦发现读写操作的会话密钥不合法，则终止会话.

会话建立是在身份验证阶段. 这里采用了挑战-应答协议，一种经典的密码学身份验证方法，其安全性得到了广泛的研究. 挑战-应答协议保证了身份验证的安全性，因为：(1) 用户密钥没有被明文传输，不会被泄露；(2) 每次用户认证的应答由随机数决定，不受重放攻击影响；(3) 会话密钥是加密的，并只有用户可以解密. 这样，身份验证阶段结束时，只有用户和可信固态硬盘了解会话密钥，保证了读写操作可以在会话中安全地进行.

会话建立之后就进入授权读写阶段. 读写操作的验证，是以其附带的操作令牌(token)为依据的. 一个最直接的设计是把会话密钥作为令牌，只要查找会话表(图 1(a))就可以验证操作附带的会话密钥是否正确，从而完成操作的验证. 但这样并不安全，因为任何可以监听用户进程与可信固态硬盘交互的进程都可以窃取会话密钥，从而利用会话密钥冒充合法用户. 最终的设计是令操作令牌

$$\text{token}=\text{MAC}_{\text{skey}}(\text{lpn},\text{num\_sectors},\text{sid})$$

即操作令牌是用户操作命令的消息认证码(Message Authentication Code，MAC). 这样有 3 个好处：(1) 由于操作令牌的计算需要会话密钥(skey)，因此只有合法用户的操作才有合法的令牌；(2) 根据消息认证码的性质，攻击者是无法通过令牌的值推测出会话密钥的值的；(3) 攻击者无法篡改用户发出的合法命令.

从上面的分析可以看出，可信固态硬盘的安全协议保证只有合法用户才能被授权执行的合法操作，并且在这个过程中不泄露任何用户密钥的信息，即使攻击者(如拥有特权的外部黑客者或内部管理员)可以监听所有操作.

即使设计再周全，如果实现有漏洞，那么最终系统也是不安全的. 虽然可信固态硬盘中的固件程序也具有相当的复杂性(约 1 万行 C 代码)，但相比操作系统(Linux 的源代码有约 1500 万行)和关系型数据管理系统库(MySQL 的源代码有约 150 万行)，可信固态硬盘中的固件程序出现软件错误或安全漏洞的可能性大大低于这些大型系统软件. 较小的代码量使得深入的代码检查，或形式验证(Formal Verification)更可行[22]. 除了担忧固件程序的正确性外，用户可能还对信任可信固态硬盘的厂商有疑虑. 这方面可以效法密码协处理器的做法，由可信权威的第三方机构评估产品的安全性，并对合格的产品授予证书[23]，打消或缓解用户的疑虑.

综上所述，可信固态硬盘为数据提供安全、有效和可靠的隐私性保护，有望成为大数据安全的新基础.

3.4　应用举例

本小节以 HDFS(Hadoop Distributed File System)为例子简要介绍可信固态硬盘如何提高大数据系统的数据安全性.

HDFS 的现有安全机制需要名字节点和数据节点协作保证(如图 2 左边). 名字节点首先验证 HDFS 客户端发来的文件操作(比如读操作)是否来自一个权限足够的用户. 核实用户身份的凭证是其提供的代理令牌(Delegation token). 如果用户合法、权限足够，则把操作所要块的块令牌(Block token)返回给客户端，这个块令牌是加密过的，无法

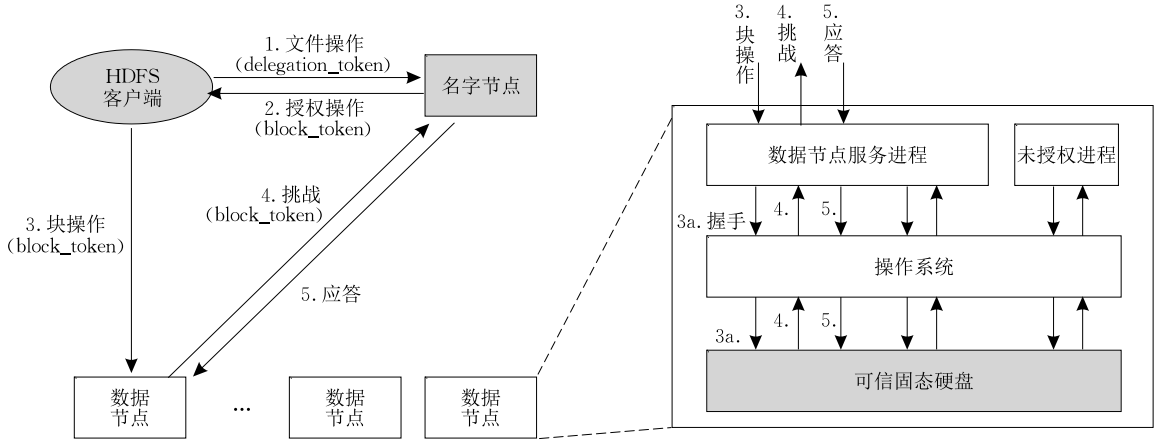


图 2　基于可信固态硬盘的安全增强型 HDFS

伪造,且过期即失效.随后客户端向数据节点调用块操作时,会把块令牌也发送给数据节点,后者检查块令牌的合法性.如果合法,才授权客户端的块操作.以上就是 HDFS 访问控制的基本原理.

但上面的访问控制是有不足的.HDFS 块是在数据节点上以文件的方式存放的.这意味数据节点的内部管理员或获得特权的外部攻击者,可以绕开上述的 HDFS 访问控制机制,直接读取块所在的文件.可信固态硬盘可以防止这种情况发生.用户的可信固态硬盘秘钥可以存储在名字节点(认为是可信的),当他需要读取受可信固态硬盘保护的 HDFS 块时,按照 3.2 和 3.3 小节所描述的协议完成身份验证和操作授权即可(如图 2 右边);即使是拥有特权的攻击者,没有用户的可信固态硬盘秘钥,也无法从可信固态硬盘中非法读取数据.

4 设计与实现

可信固态硬盘的原型系统主要包括固态硬盘的固件程序、操作系统的修改和用户库函数 3 个部分.在设计和实现原型系统的过程中,本工作主要解决了 4 个挑战:

(1)提高固态硬盘的基准性能.可信固态硬盘是基于 OpenSSD Jasmine 平台①开发,而后者已有的开源固件实现的是普通的页映射闪存转换层,存在写放大(Write amplification)、并行不充分等问题,因而性能不佳.

(2)设计高效的访问控制机制.块设备最小的读写单元是扇区(512 个字节),因此访问控制应精确到扇区,即为每个扇区都记录权限信息.然而,OpenSSD 硬件支持的带外区域(Out-of-Band Area)除了纠错码(Error Correction Code)外,无可用于记录权限信息的空闲空间.最直接的解决方法是用一个权限表,但表的大小很大,会导致很多缓存缺失,严重影响性能.

(3)修改复杂的操作系统 I/O 栈.Linux 操作系统的 I/O 软件栈,包括系统调用、虚拟文件系统、缓冲区、通用块设备层和 SCSI 驱动程序层(其中又包括高层、中层和底层驱动)等等,是一个包含诸多组件的多层架构.这种结构虽然具有模块化、松耦合和灵活性等优点,但用户传递参数(如会话 ID 和操作令牌)到固态硬盘需要经过所有这些层次.尽量少改动就要达到目的,是很有挑战性的.

(4)扩展现有的硬盘命令集.软件的数据结构容易扩展,硬件的物理接口却难以改动了.硬盘的物理接口、协议和命令集等已经形成了若干既定的行业标准.在与现有硬盘接口标准兼容的情况下,支持可信固态硬盘的接口协议所需的新命令(如挑战和应答)也是一个挑战.

这 4 项挑战的解决方法接下来将分别在"闪存转换层"、"访问控制层"、"操作系统 I/O 栈"和"硬盘接口"等 4 个小节中阐述.

在详细介绍可信固态硬盘的设计与实现之前,有必要对 OpenSSD 固态硬盘开发平台做一个简略的介绍.OpenSSD Jasmine 开发平台是一个开放的固态硬盘参考实现,基于 Indilinx 公司的 Barefoot™ 控制器,后者已经成功应用于众多厂商的固态硬盘产品.OpenSSD Jasmine 开发板,包括基于 ARM7 处理器的闪存控制器;96 KB 的 SRAM,作为程序的内存空间,存放程序和数据;64 MB 的 DRAM,主要作为 I/O 缓冲区;8 个 MLC NAND 闪存模块插口,至多达到 256 GB 闪存容量.

4.1 闪存转换层(FTL)

闪存转换层(Flash Translation Layer,FTL)是基于 NAND 闪存的固态硬盘的核心控制逻辑.FTL 将来自上层的块设备命令(读、写操作)转换为对下层的 NAND 闪存单元的命令(读、写和擦除操作).这种转换的必要性是源自因 NAND 闪存的物理特性而导致的读写不对称:闪存的读写操作均以页为单位(OpenSSD 的页大小是 32 KiB),但写操作只能对空白页,非空白页必须擦除后才能写,而擦除比读写操作的时间长得多,且是以块为单位的(OpenSSD 的块大小是 4 MiB,一个块包含 128 个页).因此,固态硬盘放弃了传统硬盘的原地更新方式,而是采用异地更新(Out-of-Place Update)的策略,即要对一个逻辑页(或者逻辑块)更新,首先在一个空白页(块)写新数据,然后把旧数据所在的页标记为无效,最后更新映射表以反映逻辑页(块)与物理页(块)之间对应关系的变化,至此完成更新操作.

FTL 的实现,根据映射表的颗粒度,可以分为页级 FTL[24-25]、块级 FTL[26]和混合型 FTL[27-28].目前学术界普遍看法是这 3 种类型的 FTL 中页级映射是性能最好的.顾名思义,页级 FTL 允许逻辑页

① The Open SSD Project. http://www.openssd-project.org/wiki/The_OpenSSD_Project

地址(Logical Page Number，LPN)可以直接转换成物理页地址(Physical Page Number，PPN). 为此，页级 FTL 需要维护一张页级映射表，为每个逻辑页都需要保存一条记录；整张表都要持久地保存在闪存中，正在使用的部分载入到 DRAM 中缓存起来，以加速读取. 这张表的大小往往超出固态硬盘上的 DRAM 容量，但由于实际负载一般都具有较好的局部性，缓存的命中率一般都很高.

OpenSSD 开源固件程序实现了页级 FTL，但性能不佳. OpenSSD开发板装有 128GB 闪存，写操作最高能达到 180 MB/s 的吞吐量，读能达到 200 MB/s. 然而，对常见的 4 KiB① 随机读写负载，性能则只有 10 MB/s 左右. 最终发现性能不佳的根源是写扩大(Write Amplification)和并行不充分.

(1) 写扩大

考虑更新一个扇区(512 B)的操作，根据异地更新的原理，FTL 必须首先把这个扇区原来所在的物理页读出到内存中的一个页缓冲区，然后更新这个缓冲区相应扇区，最后把这个页缓冲区写回到闪存，并更新映射表. 这样，更新一个扇区的开销等于一次页读加一次页写，即为了更新 512 B 的数据不得不先闪存读 32 KiB 再和写 32 KiB(OpenSSD 的一个物理页是 32 KiB)，这就是“写扩大”.

为了解决写扩大的问题，可信固态硬盘改进的 FTL 引入了子页级映射和归并缓冲区两个技术. 子页级映射，把一个 32 KiB 的页划分为 8 个 4 KiB 的区域，后者被称为子页(Sub-page). 映射表负责把逻辑子页(Logical Sub-page)对应到物理页. 引入子页之后，更新数据的最小单位变为了子页，但毕竟最终写入闪存时还必须以页为单位，因此又引入了归并缓冲区，可以把多个独立的逻辑子页更新操作合并为一次物理页写入操作. 同时，多个归并缓冲区一起也可以起到缓存的作用，减少不必要的闪存操作. 如图 3 所示，引入子页级映射和归并缓冲区之后，更新一个扇区并不需要立刻按页写回，而是首先写入归并缓冲区，这样一次物理页写回至少有 8 次子页更新分摊，因而更新一个扇区相当于只需要写 1/8 个物理页；也就是说，优化后的效果是更新 512 B 数据的开销是闪存读② 4 KiB 加闪存写 4 KiB 数据，只有原开销(32 KiB 先读再写)的 1/8.

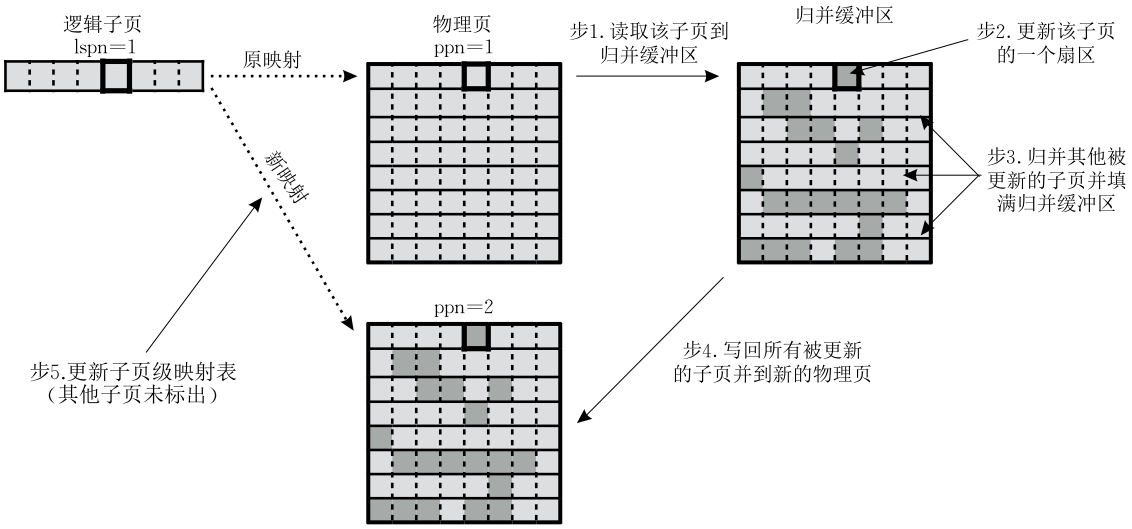


图 3　子页级映射和归并缓冲区(图 3 展示了在使用子页级映射和加入归并缓冲区之后闪存转换层更新一个扇区的流程. 图中虚线表示逻辑子页到物理页的映射关系，实线表示流程中的每个步骤，小虚线格子表示扇区，实粗线格子表示要更新的扇区. lspn 表示逻辑子页序号，ppn 表示物理页序号. 浅灰色格子表示未更新的数据，深灰色格子表示更新的数据. 更新前，逻辑子页 1 存储于物理页 1；更新后，于物理页 2. 由图可见，更新一个扇区相当于需要先读再写 4 KiB 数据)

(2) 并行不充分

固态硬盘的一大优点是其吞吐量可以随闪存模块数量的扩充而线性增加. 这要得益于其内部丰富的硬件并行性. 以 OpenSSD Jasmine 开发板为例，它拥有 4 个完全独立的通道(Channel)，每个通道有最多 8 个独立的库(Bank)；同一通道的各个库，在任意时刻只能有一个接收闪存指令，但可以各自独立地执行其已经接收到的闪存命令(读、写和擦出). 因此，固件程序应该尽量做好调度工作，充分发挥这 32 个库的并行性.

① 注意 1 KiB=1024 Bit，1 K=1000 Bit. 在 FTL 中的页和块等的大小都是扇区的整数倍，为了强调这点时会用 KiB 作为单位.

② 闪存读的单位都是扇区，而不是物理页，因此可以读部分物理页.

然而，FTL 实现中不可避免地要遇到等待 I/O 完成的情况. 比如，物理页写回之前，必须等待物理页的未更改部分从闪存中读出. 再如，当映射表中某个记录缓存缺失时，必须先等需要的条目从闪存载入到内存. 在 OpenSSD 原 FTL 实现中，CPU 需要忙等在这些阻塞的闪存 I/O 完成；在完成前，无法接收新的块设备命令或向闲置的库发送新的闪存命令，因而硬件的并行性也没有得到充分利用.

提高固件程序并行性的根本解决办法是多线程编程. 然而，开发固件是在嵌入式编程环境，直接面向底层硬件，没有操作系统，当然也没有线程这种操作系统提供的功能. 一个最直接的想法是引入为嵌入式系统而设计的实时操作系统(Real-Time Operating System，RTOS)，后者可以提供轻量级的进程，以及高效的进程调度. 但 OpenSSD 的 SRAM 只有 96 KB，容纳现有的固件程序已然稍显不足，很难再支撑一个操作系统. 引入现成的多线程支持不行，只好自己动手，自主实现了进程的上下文切换、进程调度和进程间通讯等一系列与多线程编程有关的、原本由操作系统提供的编程原语. 引入多线程后的一大挑战是保证乱序执行的用户请求的最终结果保持语义的正确性. 为此，又引入了锁机制，并在并发编程时小心谨慎地使用互斥和共享锁，在语义正确的前提下最大化各种硬件资源的使用率.

4.2 访问控制层(ACL)

访问控制层(Access Control Layer，后简称 ACL)是可信固态硬盘的安全引擎，与 FTL 紧密结合，确保用户请求被安全地接收和执行. ACL 主要功能有两个：一是用户验证，主要涉及到的数据结构是用户表；二是操作授权，主要用到的数据结构是会话表和权限表(见图 1(a)). 用户验证的原理已经在 3.2 节中介绍，具体实现中涉及到加密和安全伪随机数生成等常见密码学操作都采用了广泛验证的做法，在此不赘述. 固态硬盘的每秒输入输出数(IOPS)可能达到 10000 以上，操作授权伴随每一个用户读写请求. 因此，高效的操作授权对达到可信固态硬盘的“高安全保证、低运行开销”的目标至关重要. 本小节下面介绍如何设计和实现高效的操作授权.

操作授权首先要高效地验证操作密令. 3.3 小节给出了一个安全的操作令牌设计，即使用操作的消息认证码(Message Authentication Code，MAC). 常见的 MAC 实现依赖于块加密算法或安全哈希函数，计算开销较高. 为了在保证安全性的前提下尽量降低开销，采用了 UMAC[29]，它被认为是最快的 MAC 算法，软件实现的峰值速度可以到约 1 时钟周期/字节.

操作授权还要高效地查看或修改操作所涉及页面的权限(这里权限主要指数据的拥有者). 由于块设备的最小访问单位是扇区，原则上可信固态硬盘应该为每个扇区维护权限信息. 闪存上的每个扇区通常会留一些额外的空间存储系统信息，这个额外空间被称为带外区域. 受 OpenSSD 的硬件限制，带外区域仅能用于存放纠错码，没有额外的空间可供存放权限信息. 所以，只能在闪存上专门分配一些空间以保存权限信息，即权限表. 对于 128 GB 的固态硬盘，扇区数高达 256 M，假设每个扇区的权限信息是 2 个字节，那么权限表的大小达到 512 MB；相比之下，对于同等大小的固态硬盘，页映射表也才 128 MB. OpenSSD 拥有 64 MB 的 DRAM，其中只有不到 20 MB 可以作为页映射表和权限表的缓存. 如果采用 512 MB 权限表的设计，无疑会导致页映射表和权限表的缓存命中率大幅降低，增加很多额外的闪存读写，显著影响性能.

因此，压缩权限表的大小是很有必要的. 我们使用了下面两项技术：4 KiB 对齐的文件系统，和用户觉察的归并缓冲区.

如果所有磁盘请求都是 4 KiB 对齐的(起始地址和请求大小都是 4 KiB 的整数倍)，那么就不用以扇区为单元记录权限信息，而是以 4 KiB 的子页为单位，这样权限表的大小就降到原来的 1/8. 而幸运的是，让(几乎)所有磁盘请求都为 4 KiB 对齐是容易办到的，只要(1)在为可信固态硬盘分区的时候指定分区都以 4 KiB 对齐，比如常见的分区工具 fdisk 就支持这个功能；(2)在为可信固态硬盘上的分区创建文件系统的时候指定文件系统的块大小是 4 KiB(或 4 KiB 的倍数)，比如 Linux 的主流文件系统 Ext4 就可以设置块大小. 这样，用户的应用程序经由文件系统向可信固态硬盘发出的磁盘请求就是以 4 KiB 对齐的了. 注意，ACL 利用 4 KiB 对齐文件系统的特点减少权限表的大小，但 ACL 的安全性不依赖这点，即非 4 KiB 对齐的磁盘请求也可以安全地处理.

为了进一步压缩权限表的大小，ACL 引入用户察觉的归并缓冲区. 4.1 小节介绍了归并缓冲区如何解决写放大的问题，以及多个归并缓冲区组成读

写缓存的好处. 与之前的缓冲区不同的是，一个用户察觉的归并缓冲区只能归并属于同一个用户的子页. 这个性质可以保证由用户察觉的归并缓冲区写回到的物理页中都是同一个用户的数据. 既然如此，ACL 只需以物理页为单位记录权限信息. 这样，最终得到的权限表是以物理页为单位的，其大小是 8 MB，只有最初估计的 1/64，完全可以全部缓存在 DRAM 中.

综上所述，可信固态硬盘的访问控制机制经过精心设计后是可以高效实现的.

4.3　操作系统 I/O 栈

可信固态硬盘完全与普通硬盘兼容；这意味着用户程序、操作系统和驱动程序等无需任何改动，就把可信固态硬盘作为普通硬盘正常使用. 但如果用户希望读取可信固态硬盘中受保护的数据，或向可信固态硬盘写入受保护的数据，则需要有操作系统的支持. 需要指出的是，虽然可信固态硬盘的功能完整性依赖于操作系统，但其安全性是独立于主机软硬件的.

为了读取文件中受可信固态硬盘保护的数据，或向文件中写入受保护的数据，除了操作系统的支持外，用户程序还要使用可信固态硬盘提供的 API，其中最重要的两个函数(C 语言)是：

```
int tssd_open(const char* path, int flags);
int tssd_use_token(int fd, tssd_token_t token);
```

tssd_open 是可信固态硬盘提供的文件打开函数，其特别之处在于它强制以直接 I/O(Direct I/O)模式打开文件. 直接 I/O 模式常用于自主管理磁盘缓存、无需操作系统缓存的数据密集型应用程序，如数据库系统. 操作系统高速磁盘缓存使得其中缓存的页可以被多个不同的进程共享复用，省去很多磁盘 I/O. 但这个高速磁盘缓存的优点对于访问控制却是安全隐患，因为非可信固态硬盘授权的用户有可能从操作系统的磁盘缓存读取到属于其他用户的数据. 以直接 I/O 模式，跳过操作系统磁盘缓存，可以避免上述问题. 值得指出的是，对数据库系统和 Hadoop 这样的应用，直接 I/O 模式并不会因绕过操作系统缓存而降低性能，因为数据库系统本身维护了独立于操作系统的缓存，而 Hadoop 的主要读写模式是顺序的，缓存作用非常有限.

tssd_use_token 是用于向可信固态硬盘传递本次读写请求的操作令牌的函数. 操作令牌是可信固态硬盘授权每个读写请求的凭据. Linux 操作系统的 I/O 软件栈，包括虚拟文件系统、磁盘缓冲、通用块设备层、SCSI 驱动程序(其中又包括高层、中层和底层驱动)等等，是一个包含诸多组件的多层架构. 为了把操作令牌传递到最底层的可信固态硬盘，我们修改了 Linux 操作系统 I/O 子系统和 SCSI 子系统中各个有关组件，并扩充了它们之间通信的数据结构(如图 4).

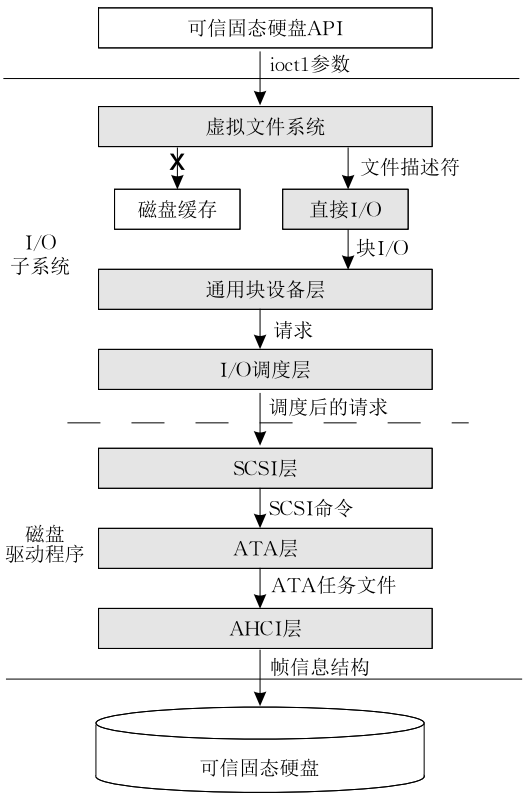


图 4　修改的操作系统 I/O 栈(图中展示了为向可信固态硬盘传递额外参数(如会话 ID 和操作令牌)而需要修改到操作系统 I/O 组件. 图中灰色方框是修改的组))

4.4　硬件接口扩展

软件的数据结构容易扩展，硬件的物理接口却难以改动了. 硬盘的物理接口、协议和命令集等已经形成了若干既定的行业标准，如果可信固态硬盘的接口无法与现有标准兼容，那么必然会提高生产商的成本，增加客户采纳的难度，最终难以推广. 幸运的是，当今使用最广泛的硬盘接口标准，如 SATA(OpenSSD 的接口)和 SAS 等，都是基于 SCSI 命令集，而 SCSI 标准具有一定的扩展空间. 可信固态硬盘对硬盘接口命令的扩充包括两种形式：

(1) 定义新的命令. 为完成基于挑战-应答的用户身份验证，可信固态硬盘的接口需要支持两个额外的命令. 第 1 个是挑战命令，定义为 sid，rnd ← Challenge(uid)，其中参数 uid 是请求建立会话的用

户 ID,返回值 sid 是即将建立的会话的 ID,rnd 是一个随机数,即所谓的“挑战”.第 2 个是应答命令,定义为 Response(sid,resp),其中参数 sid 是即将建立的会话 ID,resp 是对随机数 rnd 的应答(详见 3.2 小节).SCSI 支持最多 255 种固定长度命令以及最多 255 种变长命令,而标准已定义的命令不超过 100 种,因此 Challenge 和 Response 可以作为厂商特定功能(Vendor-specific feature)的新命令而加入.

(2)扩展现有命令.可信固态硬盘的授权读写要求传入会话 ID 和会话秘钥等两个额外的参数,即 sid 和 token(详见 3.2 小节).也就是说,可信固态硬盘的读命令为 data←Read(lpn,num_sector[,sid=0,token=0]),写命令扩展为 Write(lpn,num_sector,data[,sid=0,token=0]),其中的 sid 和 token 都是可选参数,缺省值是 0.原则上来说,可信固态硬盘具有访问控制机制的读/写命令也是可以作为新命令加入的;但实现上是不允许这样做的,因为 OpenSSD 的 SATA 控制器对 SATA 的读写命令有特殊的处理(比如有硬件实现的请求队列,还有数据缓冲区的支持),而这些硬件控制逻辑无法通过固件程序的更改.因此,只能扩展普通的读写命令,并把 sid 和 token 作为可选参数.AHCI 层向 SATA 接口的固态硬盘发送各种 SATA 命令的数据格式是依据一种叫帧信息结构(Frame Information Structure)的 ATA 标准[30].在仔细研究 ATA 标准之后,发现帧信息结构有 40 个未用或保留的比特可以使用.通过复用帧信息结构已有字段的闲置空间,得以把 sid 和 token 等信息随读写命令的其他字段一起发送给可信固态硬盘.

由上可见,可信固态硬盘需要支持的新命令可以通过在现有硬盘接口标准之上扩展而实现,并保持与现有标准兼容.

5 实验结果

本节给出对可信固态盘的初步实验结果.实验环境是一台 Intel 2.4GHz 四核处理器、8GB 内存的服务器,运行 64 位 Ubuntu 12.04 的 Linux 操作系统.可信固态硬盘(OpenSSD 开发板)与主机通过 SATA 2.0 接口连接.实验的主要目的是评估在固态硬盘中加入额外的安全机制对整体性能的影响.为此,分别在合成负载、真实负载和 HDFS 上做了测试.

5.1 合成负载的实验结果

合成的负载,按操作类型分为读请求和写请求;按请求块大小可分为 2KB、4KB、8KB、16KB、32KB 和 64KB;按访问模式可分为:顺序和随机.为了完全控制负载的模式,编写了专用的测试工具用于生成合成负载.以 4KB 随机读写负载为例,测试工具发送给固态硬盘的 I/O 请求都是读请求,且每个读请求是 4KB 大小,起始地址是随机的.

图 5 中展示了在合成负载上的结果,可以看出:(1)可信固态硬盘大幅提高了原 OpenSSD 固件的性能,尤其是随机读写的情况,最高达到了近 10 倍的性能提升(对 32KB 随机写的情况);(2)可信固态硬盘的访问控制只增加了很小的开销.这说明在第 4 节“设计与实现”中介绍对 FTL 和 ACL 的优化是有效的.

5.2 真实负载的实验结果

为了进一步验证对 ACL 优化的效果,又在真实负载上做了实验.真实负载是靠重现收集自实际生产环境的存储访问日志来产生的.从 SNIA① 和 UMASS② 等两个公开的真实存储访问记录库中选择了 Exchange、BuildServer、Financial、TPC-C 和 WebSearch 等 5 个特点不同、有代表性的存储访问日志(负载特点见表 1).我们编写了专门的测试工具用于解析和重现这些负载记录.为了更加贴近真实的使用情景,在真实负载的实验中,模拟多个不同用户(8 个用户)同时登陆可信固态硬盘的情况.

表 1 中比较了平凡 ACL 和优化 ACL 相对于禁用 ACL 的可信固态硬盘的性能.可以看到,平凡 ACL 会带来显著的开销,而优化后的 ACL(见 4.2 节“访问控制层”)的开销只有不到 3%.由此可见,我们对 ACL 的优化是非常有效的.

5.3 HDFS 的实验结果

在不牺牲性能的条件下,为大数据系统中的海量数据提供更加安全的存储服务,是本课题的目标.3.4 节介绍了一种基于可信固态硬盘的安全增强型 HDFS.根据这一架构设计,我们扩展了 HDFS 的功能,在名字节点实现了集群环境下可信固态硬盘所需的用户密钥管理和分发功能,在数据节点上实现了基于可信固态硬盘的用户验证和操作授权.

HDFS 的实验集群包括一台名字节点和一台数据节点,数据节点的服务器直连一块可信固态硬盘,

① SNIA. IOTTA Repository Home. http://iotta.snia.org/
② UMass Trace Repository. http://traces.cs.umass.edu/index.php/Storage/Storage

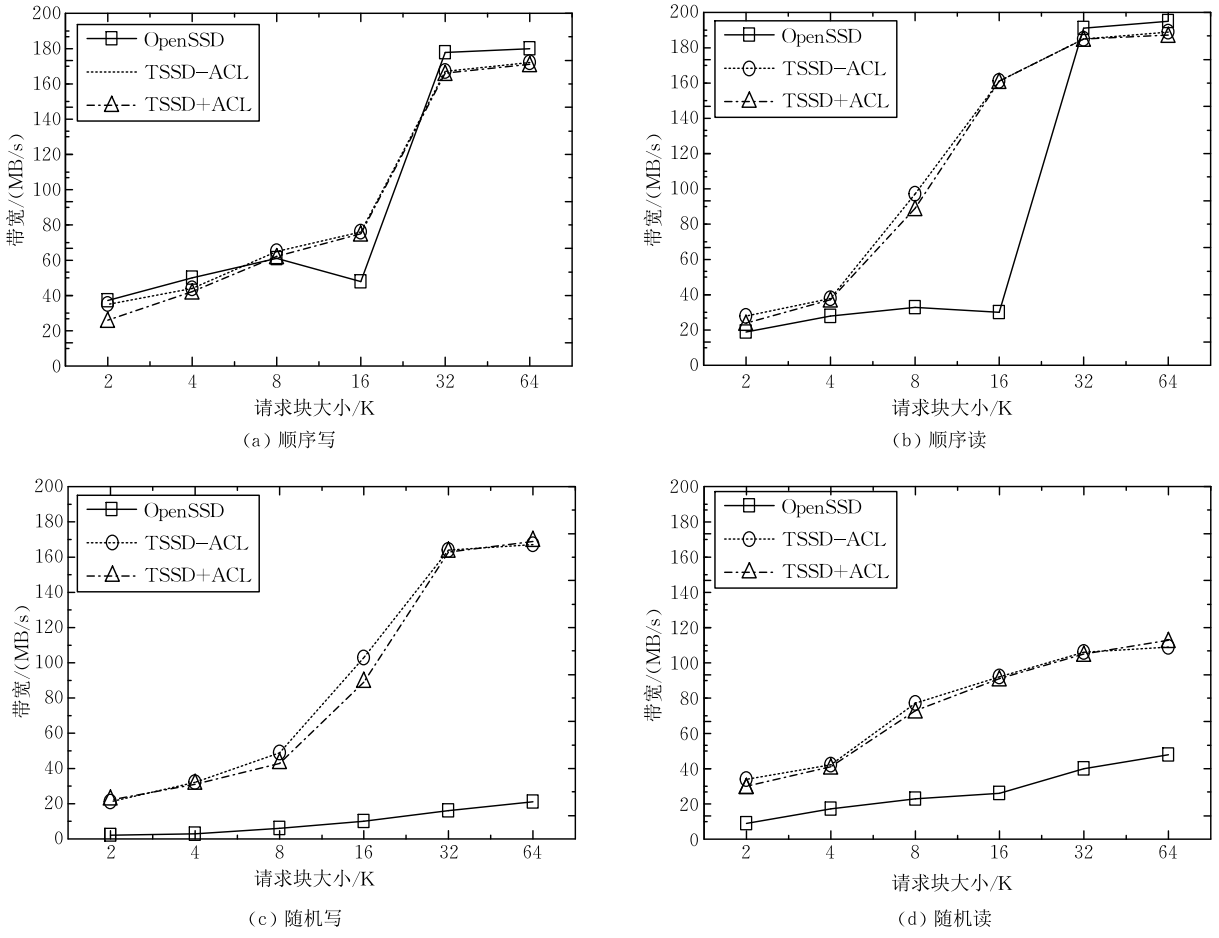


图 5　合成负载的测试结果(图 5 比较了 3 个固件实现：OpenSSD 表示自带的开源固件，TSSD-ACL 表示可信固态硬盘的固件但禁用访问控制层(ACL)，TSSD+ACL 表示可信固态硬盘的固件(带访问控制))

表 1　真实负载的测试结果(表中展示了在多种不同特点的真实负载下，可信固态硬盘的访问控制层对性能的影响.具体来说，它比较了无访问控制层(TSSD−ACL)、平凡的访问控制实现(TSSD+平凡的ACL)和优化的访问控制层制(TSSD+优化的ACL).响应时间是操作系统向可信固态硬盘发送 I/O 请求的平均响应时间，额外开销是以无访问控制的可信固态硬盘为基准)

负载名称	负载特征		TSSD − ACL(性能基准)		TSSD + 平凡的 ACL		TSSD + 优化的 ACL	
	读比例/%	平均块大小/KB	响应时间/μs	额外开销	响应时间/μs	额外开销/%	响应时间/μs	额外开销/%
Exchange	6.80	24	365	N/A	409	12	375	**2.7**
BuildServer	8.36	37	451	N/A	640	42	461	**2.2**
Financial	15.40	3	179	N/A	326	82	184	**2.8**
TPC-C	58.44	9	243	N/A	411	69	246	**1.2**
WebSearch	99.99	15	332	N/A	485	46	335	**0.9**

作为其上 HDFS 数据的存储设备.实验目标是测试基于可信固态硬盘的安全增强型 HDFS 的性能，测试工具是 DFSIO 基准测试[31]，它专门用于测量 HDFS 集群的平均读写带宽.实验中，读或写的单个文件大小设为 4 GB，文件数量是 8 个，且对于可信固态硬盘来说这些文件都属于同一个用户拥有.HDFS 集群的副本因子(Replication Factor)设为 1.

表 2 中可以看出基于可信固态硬盘的额外安全机制对 HDFS 的性能影响很小.由于只拥有一块 OpenSSD 开发板，我们不具备在更多的数据节点上进行实验的条件.但这并不影响实验数据的代表性和有效性，因为以可扩展性强著称的 HDFS 在单个节点上的吞吐量并不受到集群节点数量的显著影响[31].

表 2　HDFS 的实验结果

	原 HDFS	基于可信固态硬盘的安全加强型 HDFS	
	吞吐量/(MB/s)	吞吐量/(MB/s)	额外开销/%
顺序写	137	133.0	3
顺序读	158	155.0	2
随机读	93	91.4	2

综合合成负载、真实负载和 HDFS 等的实验结果可见,可信固态硬盘的安全机制仅产生了很小的运行开销.

6 结 论

本文提出一种解决大数据平台数据存储安全的新思路,存储内安全(In-Storage Security)把对数据的访问控制从主机上的系统软件下放到底层存储内部.更具体地说,我们设计并开发可信固态硬盘(TrustedSSD),它提供安全增强的存储设备接口和协议,使得用户可以对存储中的数据施以细粒度的访问控制,从而保障存储中数据的安全.文中深入分析了可信固态硬盘的安全性,并详细介绍了系统设计与实现中的挑战和应对.实验结果表明,无论是在合成的、还是真实的工作负载上,可信固态硬盘的运行开销不到 3%.因此,我们认为可信固态硬盘有望成为大数据安全的新基础.

参 考 文 献

[1] O'Malley O, Zhang K, Radia S, et al. Hadoop security design. Sunnyvale, USA: Yahoo Inc., Tech Report, 2009

[2] Kim S, Oh H, Park C, et al. Fast, energy efficient scan inside flash memory SSDs//Proceedings of the International Workshop on Accelerating Data Management Systems (ADMS). Seattle, USA, 2011

[3] Do J, Kee Y S, Patel J M, et al. Query processing on smart SSDs: Opportunities and challenges//Proceedings of the 2013 ACM International Conference on Management of Data (SIGMOD'13). New York, USA, 2013: 1221-1230

[4] Kang Y, Kee Y, Miller E L, et al. Enabling cost-effective data processing with smart SSD//Proceedings of the 29th IEEE Symposium on Mass Storage Systems and Technologies (MSST'13). Long Beach, USA, 2013: 1-12

[5] Kang W H, Lee S W, Moon B, et al. X-FTL: Transactional FTL for SQLite databases//Proceedings of the 2013 ACM International Conference on Management of Data (SIGMOD). New York, USA, 2013: 97-108

[6] Thuraisingham B. Big data security and privacy//Proceedings of the 5th ACM Conference on Data and Application Security and Privacy. San Antonio, USA, 2015: 279-280

[7] Cormode G, Srivastava D. Anonymized data: Generation, models, usage//Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data (SIGMOD'09). Providence, USA, 2009: 1015-1018

[8] Dwork C. Differential privacy: A survey of results//Agrawal M, Du Dingzhu, Duan Zhenhua, Li Angsheng eds. Theory and applications of models of computation. LNCS 4978. Berlin Heidelberg: Springer, 2008: 1-19

[9] Popa R A, Redfield C, Zeldovich N, et al. CryptDB: Protecting confidentiality with encrypted query processing//Proceedings of the 23rd ACM Symposium on Operating Systems Principle (SOSP'11). Cascais, Portugal, 2011: 85-100

[10] Arasu A, Blanas S, Eguro K, et al. Orthogonal security with cipherbase//Proceedings of the 6th Biennial Conference on Innovative Data Systems Research (ICDE'13). Asilomar, USA, 2013: 1013-1022

[11] Stevens C E. AT Attachment 8-ATA/ATAPI Command Set (ATA 8-ACS). American National Standard Institute (ANSI). New York: Working Draft Project, Technical Report T13/1699-D, 2006

[12] Strunk J D, Goodson G R, Scheinholtz M L, et al. Self-securing storage: Protecting data in compromised system//Proceedings of the 4th Conference on Symposium on Operating System Design & Implementation (OSDI'00). San Diego, USA, 2000: 12-12

[13] Gibson G A, Nagle D F, Amiri K, et al. File server scaling with network-attached secure disks//Proceedings of the 1997 ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS 1997). Seattle, USA, 1997: 272-284

[14] Wylie J J, Bigrigg M W, Strunk J D, et al. Survivable information storage systems. Computer, 2000, 33(8): 61-68

[15] Factor M, Meth K, Naor D, et al. Object storage: The future building block for storage system//Proceedings of the IEEE Local to Global Data Interoperability-Challenges and Technologies. Sardinia, Italy, 2005: 119-123

[16] Ghemawat S, Gobioff H, Leung S T. The Google file system. ACM SIGOPS Operating Systems Review, 2003, 37(5): 29-43

[17] Riedel E, Iren S. Object storage and applications//Proceedings of the 2007 Linux Storage & Filesystem Workshop. San Jose, USA, 2007: 12-13

[18] Wei M Y C, Grupp L M, Spada F E, et al. Reliably erasing data from flash-based solid state drives//Proceedings of the 9th USENIX Conference on File and Storage Technologies (FAST'11). San Jose, USA, 2011: 8

[19] Tian Lei, Jiang Hong. Protecting flash memory SSDs against malicious I/O attacks//Proceedings of the 3rd Annual Non-Volatile Memories Workshop 2012 (NVMW'12). La Jolla, USA, 2012

[20] Butler K R B, McLaughlin S E, McDaniel P D. Non-volatile memory and disks::avenues for policy architectures//Proceedings of the 2007 ACM Workshop on Computer Security Architecture. Alexandria, USA, 2007: 77-84

[21] Butler K R B, Efstathopoulos P. U can't touch this: Block-level protection for portable storage//Proceedings of the International Workshop on Software Support for Portable Storage (IWSSPS). Grenoble, France, 2009

[22] Klein G, Elphinstone K, Heiser G, et al. seL4: Formal verification of an OS kernel//Proceedings of the ACM 22nd Symposium on Operating Systems Principles (SOSP'09). Big Sky, USA, 2009: 207-220

[23] Security requirements for cryptographic modules. National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA: FIPS PUB, Technical Report 140-2, 2001

[24] Gupta A, Kim Y, Urgaonkar B. DFTL: A flash translation layer employing demand-based selective caching of page-level address mappings//Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating systems (ASPLOS). Washington, USA, 2009: 229-240

[25] Ma D, Feng J, Li G. LazyFTL: A page-level flash translation layer optimized for NAND flash memory//Proceedings of the 2011 ACM International Conference on Management of Data (SIGMOD'11). Athens, Greece, 2011: 1-12

[26] Qin Z, Wang Y, Liu D, et al. Demand-based block-level address mapping in large-scale NAND flash storage systems//Proceedings of the 8th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS 2010). Scottsdale, USA, 2010: 173-182

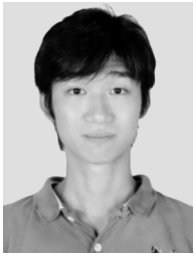
[27] Lee S W, Park D J, Chung T S, et al. A log buffer-based flash translation layer using fully-associative sector translation. ACM Transactions on Embedded Computing Systems, 2007, 6(3): 18

[28] Lee S, Shin D, Kim Y J, et al. LAST: Locality-aware sector translation for NAND flash memory-based storage systems. ACM SIGOPS Operating Systems Review, 2008, 42(6): 36-42

[29] Black J, Halevi S, Krawczyk H, et al. UMAC: Fast and secure message authentication//Proceedings of the 19th Annual International Cryptology Conference (CRYPTO'99). Santa Barbara, USA, 1999: 216-233

[30] Anderson D. SATA Storage Technology: Serial ATA. Monument, CO, USA: MindShare Press, 2007

[31] Shvachko K, Kuang H, Radia S, et al. The Hadoop distributed file system//Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST'10). Lake Tahoe, USA, 2010: 1-10



TIAN Hong-Liang, born in 1987, Ph. D. candidate. His main research interests include cloud data security, data management with new hardware.

ZHANG Yong, born in 1973, Ph. D., associate professor. His main research interests include data management and data analysis.

XU Xin-Hui, born in 1990, M. S. candidate. His main research interests focus on data management.

LI Chao, born in 1978. Ph. D., associate professor. Her main research interests include storage system and data management.

XING Chun-Xiao, born in 1967, Ph. D., professor. His main research interests include digital library and database technology.

Background

Big data, known for its characteristics of high volume, high value and centralized storage, is an attractive target for attackers. Thus, big data security is an important issue. However, the two most common data security approaches used in big data platforms (e. g. Hadoop) are not satisfactory: (1) Access control mechanisms are prone to bugs and vulnerabilities; (2) Data encryption is provably secure but incurs considerable overheads during data processing. Therefore, we believe that a fundamentally different approach is required to provide a high security guarantee for data in storage with little overhead.

Traditionally, commodity storage devices are considered untrusted. Thus, additional security mechanisms, e. g. access control and data encryption, have to be enforced by system software on the host. With the advance of storage technology like Solid State Drive (SSD), it is time to reconsider the assumption for two reasons. First, a SSD has enough spare resources for additional functionality as demonstrated by existing in-storage processing works, in which some data processing tasks are offloaded to the SSD. Second, the intrinsic component of a SSD firmware, flash translation layer (FTL), maintains a mapping table of virtual addresses from the host to physical addresses on the flash. This indirection of flash devices can be readily modified to hide the information on any part of the flash, including sensitive user data and critical device data. As a result, it is possible to introduce security mechanism into a SSD, making the commodity storage device trustworthy.

A large volume of work has been done regarding to storage security, e. g. Self-Securing Storage, Network-Attached Secure Disks, Survivable Storage System, etc. However, these works, which focus on hard-disk drives (HDD), are different from our work in terms of objective and method. To the best of our knowledge, our research is the first work that attempts to integrate security mechanism into SSD.

In this paper, we present TrustedSSD, a secure SSD that efficiently guarantees the security of data-at-rest by enforcing a fine-grained access control. We first analysis the security of TrustedSSD, and then describe the design and implementation of the system, highlighting the challenges involved. We built a prototype of TrustedSSD on a commercially successful SSD controller. Experimental results on both synthetic and real-world workloads showed that TrustedSSD incurs less than 3% overhead. Therefore, we believe TrustedSSD is a promising approach to big data security.

This work is supported by the National Basic Research Program (973 Program) of China under Grant No. 2011CB302302, the Support Program of the National '12th Five-Year-Plan' of China under Grant No. 2012AA09A408, Tsinghua University Initiative Scientific Research Program.