doi: 10.12068/j.issn.1005-3026.2019.08.004

基于 TEE 的可信存储系统设计与实现

张 强, 乔建忠
(东北大学 计算机科学与工程学院, 辽宁 沈阳 110819)

摘 要: 在对当前主流可信存储系统的分析和研究的基础上,设计并实现了符合 GP 标准,同时满足多种安全存储特性的可信存储系统(TSS). TSS 不仅能对数据进行授权加密、保证数据的完整性和一致性,同时还提供了很多其他安全存储特性(如持久存储对象的原子操作). 为了改善大数据读写性能,提出了一种在 REE 的内核空间中动态申请连续内存并通过通信管道将该连续物理内存映射到 TEE 中的方法. 这种方法可以有效地减少 TEE 和 REE 之间的切换次数、内存申请次数及内存的拷贝负载. 实验数据显示,与其他相关可信存储系统相比,TSS 有 8% 到 10% 的性能提升.

关 键 词: 可信存储;可信执行环境;GP 标准;存储特性
中图分类号: TP 311 **文献标志码**: A **文章编号**: 1005-3026(2019)08-1080-07

Design and Implement of TEE-Based Trusted Storage System

ZHANG Qiang, *QIAO Jian-zhong*
(School of Computer Science & Engineering, Northeastern University, Shenyang 110819, China. Corresponding author: ZHANG Qiang, E-mail: zqzq53373931@163.com)

Abstract: Based on the analysis of currently mainstream trusted storage system (TSS), we design and implement a trusted execution environment (TEE)-based TSS, which conforms to Global Platform (GP) standard. Our TSS provides not only authorized encryption, the integrity and consistency of data, but also many security storage operation properties such as atomicity operations of persistent object. In order to improve the read/write performance of big data, a new method is proposed for dynamically allocating continuous memory in REE's kernel memory space and mapping the address to the TEE through communication pipe. This method can reduce switching times, allocating memory times and copy memory overloads between two worlds. The experiments show that our system has an 8% to 10% performance improvement compared with related trusted storage systems.

Key words: trusted storage; trusted execution environment; GP standard; storage features

伴随着物联网(IoT)的快速发展,手机的应用已经发生了巨大变化,从原来简单的通信变成包含多种应用(娱乐、购物、支付等)的综合体. 同时在个人信息、支付信息等私有数据的存储和管理上也带来了新的挑战[1-3]. 目前有多种技术可以实现手机可信存储,如:基于云的存储、虚拟化技术、安全单元(SE),以及 TEE(trusted execution environment)等安全存储方案. 基于云的数据存储[4]需要用户将私密数据上传到第三方云存储服务中,这就要求第三方拥有较高的信任值. 手机虚拟化[5]通过隔离的方式为不同的 OS 或者进程提供运行资源,将可信存储系统运行在隔离的虚拟环境中以防止外部攻击;但虚拟化技术无硬件支持,安全性较弱,同时 Hypervisor 的可信基(trusted compute base,TCB)太大,容易出现漏洞而被攻击. SE[6]虽然提供了芯片级别防篡改能力,可以通过 SE 构建 TPM(trusted plateform module)[7-9],但是,单一芯片在计算能力上存在严重缺陷,嵌入式的内存空间无法实现较多功能;同时,SE 与其他设备的通信往往通过单一通道,

收稿日期: 2018-07-04
基金项目: 国家自然科学基金资助项目(61272177).
作者简介: 张 强(1982-),男,辽宁兴城人,东北大学博士研究生; 乔建忠(1964-),男,辽宁沈阳人,东北大学教授,博士生导师.

在时效上存在缺陷，导致 SE 只能提供较少功能接口和有限工作能力. 基于 TEE[10-11] 的可信存储方案能够保证敏感数据只能被授权的软件进行访问和处理. 使用 ARM Trusted Zone[12] 技术将一个 CPU 分成两种执行状态——可信状态（secure world，安全世界）和非可信状态（normal world，非安全世界）；可信存储系统（trusted storage system，TSS）运行在可信状态下，从而与非可信环境（例如 Android）分开. 与安全加密处理器不同，TEE 能保证：① 运行复杂的软件，如操作系统；②访问所有 normal world 中的资源，如驱动等；③通过 TEE 来控制 REE（rich execution environment）.

1　相关工作

基于 TEE 的可信存储系统主要集成在已有的 TEE 中，目前在一些院校和企业有一些研究，如表 1 所示. 密封存储能够保证私密性、完整性及数据的新鲜性，同时保证数据的授权访问；使用 TEE 的密封存储有 3 个特征：①安全加密密钥从来不会脱离 TEE；②标准加密机制；③防回滚机制（如 RPMB[8]）. 授权加密可以同时保证数据的私密性和授权访问.

表 1　基于不同 TEE 的可信存储系统特征

Table 1　Features of trusted storage system based on different TEEs

TEE 及其拥有者	可信存储特点	TEE 与 REE 通信方式
OBC（Nokia）*	存储对象采用 AES-EAX 进行加密，根密钥通过 efuse 在硬件中的加密密钥派生获得	私有接口
＜T-Base（Trustonic）*	存储单元为数据对象，并以树形结构进行存储，通过 container 来保存密钥等私密数据	未知
QSee（Qualcomm）*	未知	GP internel API & client API
OP-TEE（Linaro）	以块对象作为存储单元，通过 metadata 管理存储对象	GP internel API & client API
TLK（Nvidia）	密封存储（sealing storage）	私有接口
Open-TEE （Intel&University of Helsinki）	未知	GP internel API & client API
ANDIX OS（Graz University of Technology）	Merkle-Tree & AE；可信应用（trusted application，TA）使用安全块设备（secure block device，SBD）库实现可信存储系统	GP internel API & client API
TLR（Microsoft Research）	密封存储	.net remoting 方式的私有接口
SafeG（Nagoya University）	未知	安全 RPC
DroidVault	提供授权认证加密存储（authenticated encryption，AE）	私有接口

注：* 表示商业 TEE，其他为学术 TEE.

在 Open-TEE 中只介绍了可信存储系统，并没有给出具体设计方法和性能分析；当前 Linaro 的 OP-TEE 2.1 版本[13-14] 支持两种方式的可信存储：通过设置 CFG_REE_FS 和 CFG_RPMB_FS 编译选项，使存储对象分别存储在 REE 的文件系统和 RPMB（replay protected memory block）[8] 中；其中 RPMB 方案支持防回滚攻击功能，而另外一种不支持. 数据以块为单位进行读写和存储，过多的数据块操作会增加切换次数而影响性能. 由于 RPMB 空间有限，将所有数据对象存储在 RPMB 中，存在空间不够的问题. ANDIX OS[15] 通过给 TA 提供 SBD 库来实现可信存储，该方案使用 Merkle-Tree & AE 技术，但其假设并非每个 TA 都需要快速随机地访问存储数据；这种方案对开发者来说存在局限性. DroidVault[16] 只提供基于 AE 的可信存储，无法保证数据的新鲜性. 本文 TSS 方案的数据结构和设计与上述方案有所不同，具体如下：

1）采用密封存储，可以确保数据的保密性、完整性及授权访问，同时确保数据的原子操作；

2）使用 RPMB 机制保证数据的新鲜性，通过保存存储对象的 counter 值来防止回滚攻击；

3）每一个物理文件对应一个存储对象，简化数据格式，减少了两个世界中的切换次数；

4) 大数据对象存储特殊处理机制.

2 设计与实现

安全环境中只提供了有限的安全存储空间,无法完成所有私密数据的存储,所以必须借用 REE 的存储空间来存储私密数据,同时这种设计方式还可以减少 TEE 的 TCB. 根据图 1 所示,圆角方框代表了可信存储的主要功能模块,在安全世界中,将安全存储接口设计成为一个标准库. 可信存储的所有存储特性都在可信存储 API 库中实现. 可信存储系统(trusted storage system,TSS)提供了 RPMB、密钥服务及文件访问功能. 根据微内核的能力级访问机制,在系统启动前需要配置好所有服务的访问权限. 在非安全世界中,CA 通过 GP Client API 访问 TA 的各项功能. Andorid 系统包含三个主要设备节点(驱动程序)辅助 TEE 与 REE 之间的通信与数据访问存储,即 RPMB,Client API 及虚拟文件系统的访问. 在用户空间中的 uTDaemon 负责将通道中的数据读写到相应的存储设备中. 数据通道默认容量为 512 KB,如果传输数据量大于该值,则需要进行多次传输.

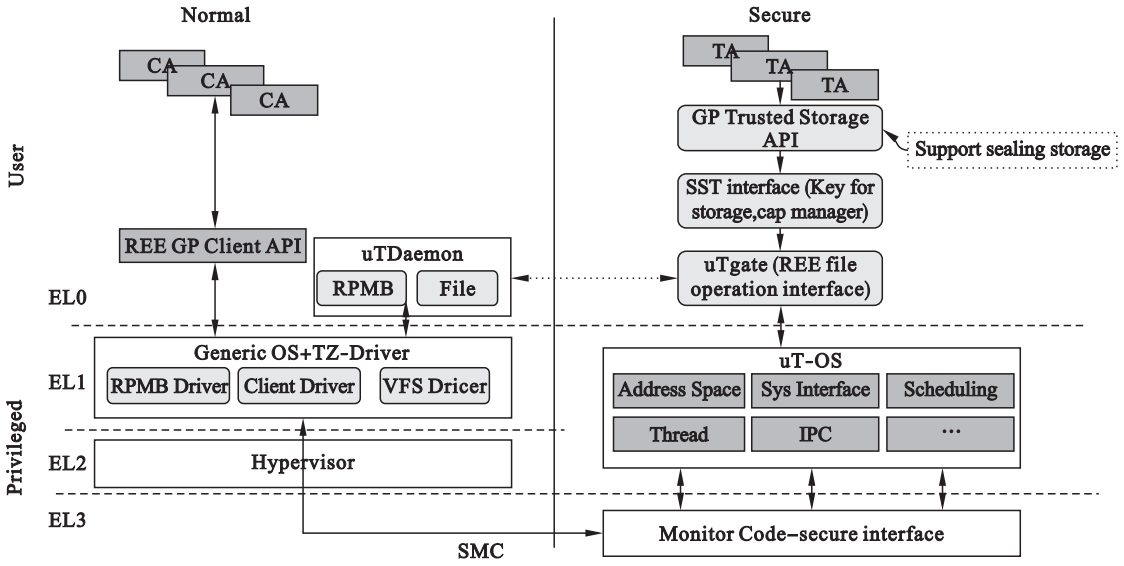

图 1 系统结构以及 TSS 功能模块

Fig. 1 TEE architecture and modules of TSS

2.1 可信存储系统的实现

2.1.1 文件存储结构及存储对象格式

TSS 通过扩展 Andorid 的文件系统来完成可信存储的数据存取,因此所有存储对象都是以加密方式存储在 REE 中. 如图 2 所示,每个 TA 都有自己的存储空间,即相应的存储路径. 所有的存储对象,如 PO1, PO2 都在指定的 UUID (universally unique identifier)路径下. 通过加密后的 ObjectID 对每个存储对象命名. 为了提高更新效率,attr_data 和 data_info 数据以块的方式存储,大小限定为 4 KB,允许存储的最大数据对象为 4 MB(4 KB × 1 024).

2.1.2 安全存储特性

1) 数据私密性. 设计中使用 AES-CTR 加密算法对数据对象进行加密.

加密密钥的使用:每个 TA 对应一个加密密钥,TSS 需要通过密钥服务来获取本子系统可以使用的密钥,这样就需要根据 TA 对当前 TSS 系统的密钥派生出每个 TA 的密钥. 根据系统需求,系统启动时通过满足 RFC 2398[17] 标准的 PBKDF2 算法将 efuse 在硬件芯片内的根密钥派生出 8 个子秘钥,分别为不同的服务及功能扩展而保留. TSS 通过 KDF 算法派生自己的加密密钥.

AES-CTR 中 IV(initialization vector)的使用:由于采用 AES-CTR 算法,因此必须考虑 IV 的设计. 为了安全考虑,设计中将 16 B 随机数(在本文设计中,16 B 的长度已经足够)用于每个加密存储对象,并将 IV 存储在明文部分. IV 的格式为{*xxxxxxxx*0*xxxxxxx*},*x* 代表随机数,所以如果前 8 个字节保持不变并采用加 1 的方法递增 IV,则可以生成最少 2^{31} 个 IV. 由于加密部分的数据长度被限定在 4 MB,这种方式一定可以满足系统设计的需求.

2) 数据完整性. TSS 通过计算存储文件 Hmac 的方法来确保数据完整性. 为了提高性能,

在打开(Open)对象时,保存该对象所有块的 Hash 值,构成 Hash 链表,读取对象时,只需要对比对象中相应块对应的 Hash 值是否一致即可,简化了完整性验证流程.

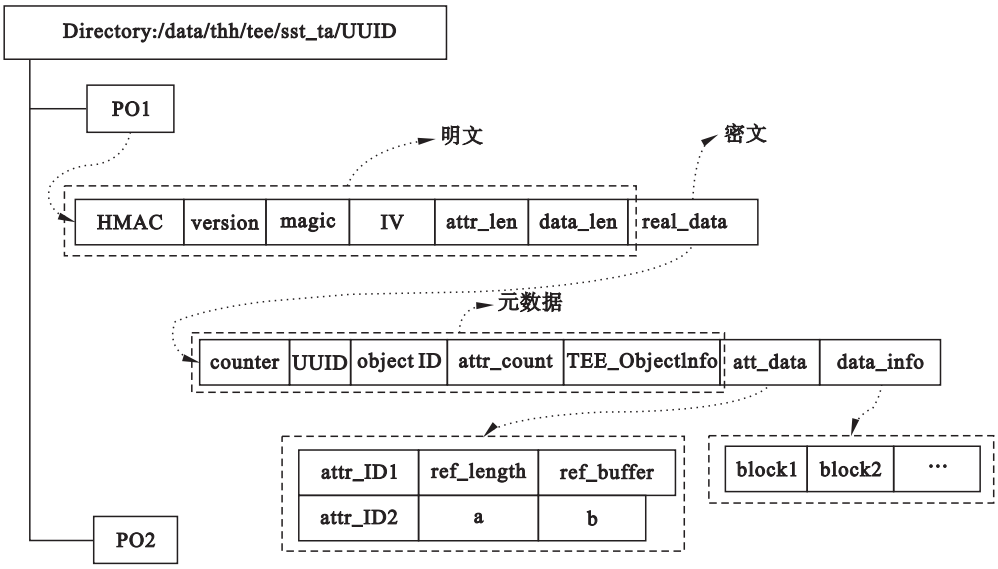

图 2　存储对象、存储路径及详细结构

Fig. 2　Detail structure of storage object and directory

3) 防回滚攻击. 为了防止攻击者用以前的数据替换当前数据对 TSS 进行回滚攻击,TSS 必须保证数据新鲜性. 为了满足该系统属性,TSS 采用 RPMB 作为辅助设备来存储对象的 ObjectID 和 counter 值,同时 counter 值只能递增. 如图 3 所示,由于系统需要,供 TSS 使用的存储空间仅为 544 KB. 如果为每个 TA 提供 18 KB 存储空间用于满足数据新鲜性,整个 TSS 系统最多可为 30 个 TA 提供存储功能.

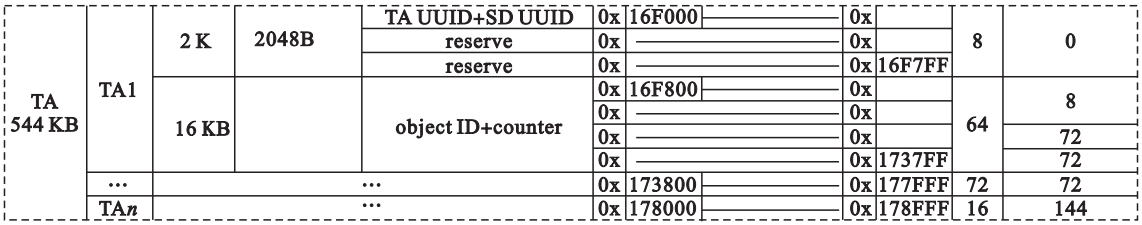

图 3　RPMB 空间布局

Fig. 3　RPMB layout

根据 GP 标准,ObjectID 的最大长度为 64 B,再加上 counter 的 4 B 长,一个 TA 最多只能存储 240 个 objectID 和 counter 值. 为了解决存储空间受限的问题,TSS 以存储 objectID 的 Hash 值的方式代替直接存储 objectID 本身,这样比原来的长度减少了一半. 具体结构如图 4 所示,其中 metadata 只有 1 B.

4) TSS 中的 API 满足原子性. 在 TSS 中,原子操作是指执行一个 API 要么成功,要么在失败时系统状态空间没有任何变化. 整个 TSS 的状态空间包含物理存储,内存信息要么变为指定的状态,要么没有改变,这对整个 TSS 提出了很多挑战. 根据 GP specification1. 1 的要求,6 个 GP API 需要满足原子性. 本文从原子执行方案和执行顺序进行分析.

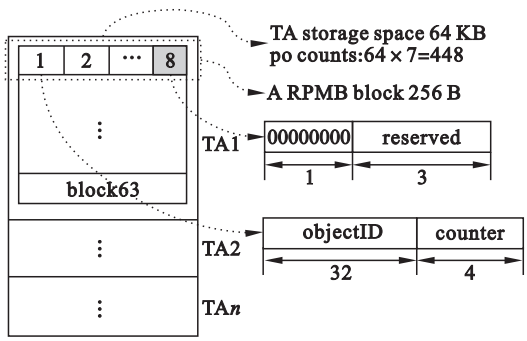

图 4　RPMB 中 objectID 和 counter 值的布局

Fig. 4　Layout of objectID and counter value in RPMB

为了满足 API 执行的原子性,对数据对象的操作存在大量交错的读(R)、写(W),以及在系统

崩溃时的恢复操作(R),其中读操作不会影响原子性. 这些操作主要包含三个部分:对内存对象及快照的写操作;物理文件及副本的写操作;对 RPMB 的写操作. 原子操作的关键是当系统处于异常或者崩溃(panic)时如何恢复到原来的状态. TSS 关注物理文件和内存空间的原子性. 对于物理文件,TSS 采用先复制后操作的方式进行操作[18],防止因 panic 而导致物理文件损坏或丢失;对于内存空间对象,TSS 采用了内存快照的方式,将要修改的内存对象复制,如果出现异常返回,直接使用快照进行恢复.

执行顺序的优化:不同的执行顺序很可能对整个 TSS 的性能造成巨大影响,在如上提到的三部分的操作中,只有 RPMB 的操作是基于硬件的,所以将其放到最后进行更新,这样就能确保前两者更新成功后,RPMB 的更新不会导致不可回退的情况发生. 如下,本文给出了相应的执行顺序和操作内容,相关详细操作对应文献[18]中的相关操作.

①执行原子操作的 API,将要修改的内存对象复制成为快照(例如:RPMB 信息、ObjectID 链表等),并对内存快照进行修改,如果失败,返回指定的错误编码(如:TEE_PANIC 等)并释放内存快照;如果成功,则继续执行.

②对当前物理文件进行拷贝,并根据预先组织的内存缓存进行更新(W1 ~ W6),但是与算法[18]不同的是不执行 W7. 如果这个过程出现 panic,则执行恢复原文件操作 R1 ~ R3,回退到原来的物理文件.

③更新 RPMB 块信息,如果更新成功,删除原来复制的物理文件并释放写文件的锁(W7,W8),根据快照更新内存对象;如果更新失败,则删除快照信息并将文件重新命名为原来的名字. 如果删除旧文件失败,则可以忽略,因为并不会影响到原子操作流程.

2.2 性能优化方案

如图 5 所示,REE 与 TEE 的通讯方式是通过数据通道和通讯通道协同完成的,两个通道的实现方式为共享内存,共享内存的大小是固定的. TSS 中两个通道都为 512 KB. uTDaemon 和 uTGate 分别用于解析和处理不同状态下两个通道中的数据.

针对大数据对象的读写,TSS 设计了另外一种方法,即通过在 REE 中申请指定长度的内核空间连续内存,通过通讯通道将地址映射到 TEE 中,再由 TEE 完成数据的填充任务. 采用这种方案可以有效减少两个世界中的切换次数以及内存的拷贝次数.

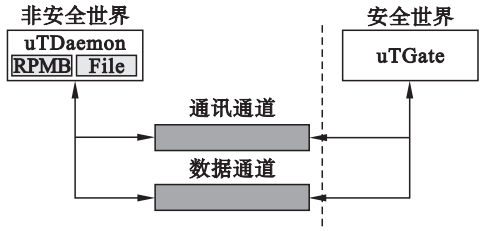

图 5 TEE 通讯通道和数据通道

Fig. 5 Communication pipe and data pipe in TEE

算法 1:普通存储对象读算法

```
current_fd ← fd
current_length ← get_len(fd, len, block_size)
buffer_block ← allocate_buff(current_length)
if current_length > pipe_size then
    while current_length > pipe_size do
        current_length - = pipe_size
        buffer_block = f_read(fd, pipe_size)
        buffer_block + = pipe_size
    end
else
    buffer_block = f_read(fd, current_length)
end
decrypt(buffer_block)
buffer ← read_buffer(buffer_block, offset)
```

根据申请内存空间时机的不同,可以使用两种策略:一种策略是在系统启动时预留一部分内存空间;另外一种策略是根据 TEE 读写文件的需求,动态地在 REE 中申请连续的内核内存空间. 第一种策略绕过了所有的内存管理方式而不受控制;第二种策略通过调用内核函数__get_free_page 申请较大内核连续内存时可能失败,但 TSS 还是选择了第二种策略,因为在申请连续内核空间失败的情况下,可以直接切换到原来的一般模式继续工作流程. 算法 2 给出了具体的工作流程.

算法 2:大数据对象读算法

```
current_fd ← fd
current_length ← get_len(fd, len, block_size)
buffer_block ← allocate_buff(current_length)
if current_length > pipe_size then
    switch to REE
    buffer_block_REE ← allocate_in_REE_kernel
    (current_length)
    if buffer_block_REE is NULL
```

```
        goto Algorithm1
    else
        pass address of buffer_block_REE to TEE
        copy buffer_block_REE to buffer_block
        release buffer_block_REE
else
    buffer_block = f_read(fd, current_length)
end
decrypt(buffer_block)
buffer ← read_buffer(buffer_block, offset)
```

3　实验分析

3.1　实验平台

TSS 的测试平台是 MTK6797/Helio X20 片上系统(Soc). 它是一款包含 4×4 核芯,并采用大小核设计架构的手机芯片. REE OS 采用 Andorid M. 由于 OP-TEE 对硬件支持的差异性,本文使用 QEMU 作为测试平台,对比 OP-TEE 中的可信存储系统. TSS 针对测试开发了专门的测试套件 MDTester,测试用例达 5 000 多个.

3.2　性能分析

3.2.1　真实硬件测试

第一个实验研究存储对象大小对 TSS 性能延迟的影响. 对测试数据取平均值(分别测试 5 000次)以减少系统其他因素带来的误差. 图 6 为随机执行 Open, Create, Read 和 Write 操作的 GP API 性能对比,可以看出,Create 和 Write 操作的时间延迟几乎一样,Write 操作的时间延迟比 Create 略有增加,这是由于在 Create 操作时,包含了 Open 操作.

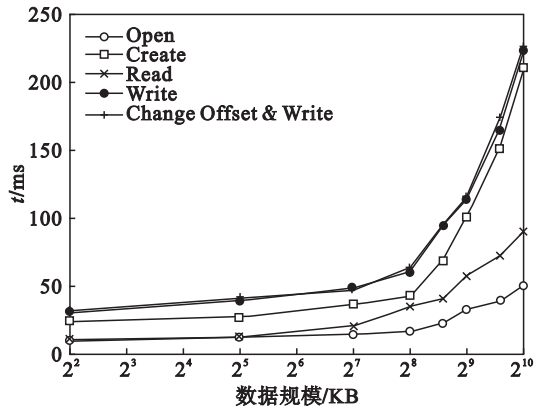

图 6　一般操作的性能对比

Fig. 6　Performance comparison of normal file operations

图 7 为使用大数据优化机制后的性能分析. 当存储数据对象小于 512 KB 时,开和关 BIG_MEM_CFG 在读写性能上几乎一致;然而,当读写的数据超过512 KB 时,读写的时间延迟有了明显变化. 例如,针对 4 096 KB 文件的读写,两种状态下的性能差距为 10% ~15% .

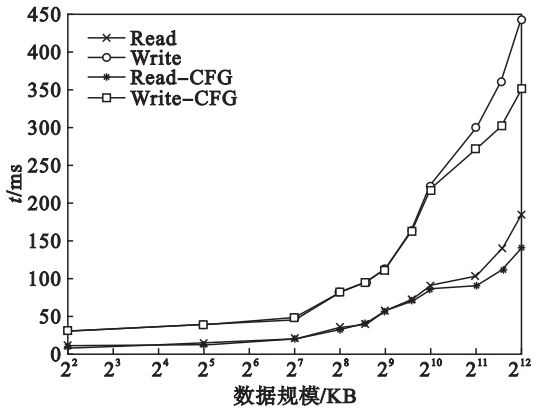

图 7　BIG_MEM_CFG 在开和关情况下的读写性能

Fig. 7　Read/write performance in the case of BIG_MEM_CFG opened and closed

3.2.2　TSS 与 OP-TEE 对比分析测试

当前的 OP-TEE 支持两种方式的文件存储:通过 CFG_RPMB_FS 预编译选项确定文件存储方式,这种方式将所有信息存储到 RPMB 中;第二种方式通过 REE 的文件系统作为扩展来存储相关信息,但这种方式没有提供数据的防回滚攻击. TSS 和 OP-TEE 都涉及到借助 REE 文件系统扩展来存储相关信息,本文以第二种存储方式进行性能对比,结果如图 8 所示. 本文存储方案比 OP-TEE 有 8% ~10% 的提升,这是由于 OP-TEE 采用简单的块存储方案,而 TSS 中针对大数据处理使用了优化策略. 虽然本文也使用块存储方式,但 OP-TEE 使用块为单位进行数据传输,这样在大数据存储的操作过程中将大大增加 REE 与 TEE 之间的切换次数,从而影响了性能;同时,OP-TEE 没有对大数据进行相应的针对性操作.

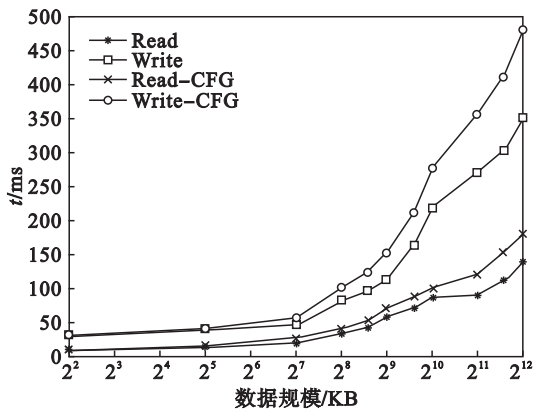

图 8　OP-TEE 与 TSS 的性能对比

Fig. 8　Performance comparison between OP-TEE and TSS

4 结 语

本文设计并实现了基于 TEE 的可信存储系统,提出了相关策略来改善可信存储的读写性能.实验表明,与当前主要以 TEE 为基础的可信存储系统相比,TSS 的设计方案在安全性上得到了提高,在数据读写的综合性能上提高了 8% ~10%.以后的工作将对 REE 端存储的安全性及 TEE 端安全存储代码本身的安全性进行分析,采用的技术涉及快速系统访问及形式化方法的设计和验证.

参考文献:

[1] The MITRE Corporation. CVE - 2015 - 4421 [EB/OL]. [2018 - 05 - 10]. http://cve. mitre. org/cgibin/cvename. cgi? name = CVE - 2015 - 4421.

[2] The MITRE Corporation. CVE - 2014 - 4322 [EB/OL]. [2018 - 05 - 10]. http://cve. mitre. org/cgi - bin/cvename. cgi? name = CVE - 2014 - 4322.

[3] The MITRE Corporation. CVE - 2015 - 4422 [EB/OL]. [2018 - 05 - 10]. http://cve. mitre. org/cgi - bin/cvename. cgi? name = CVE - 2015 - 4422.

[4] Yang H J, Costan V, Zeldovich N, et al. Authenticated storage using small trusted hardware[C] // Proceedings of the 2013 ACM Workshop on Cloud Computing Security Workshop. New York: ACM, 2013: 35 - 46.

[5] Park S W, Lim J D, Kim J N. A secure storage system for sensitive data protection based on mobile virtualization[J]. *International Journal of Distributed Sensor Networks*, 2015, 11(2): 929380.

[6] GP. GlobalPlatform made simple guide: secure element[EB/OL]. [2018 - 06 - 13]. http://www. global-platform. org/mediaguideSE. asp.

[7] Zhang X, Acıiçmez O, Seifert J P. A trusted mobile phone reference architecture via secure kernel[C] // Proceedings of the 2007 ACM Workshop on Scalable Trusted Computing. New York: ACM, 2007: 7 - 14.

[8] Winter J. Trusted computing building blocks for embedded linux-based ARM trustzone platforms[C] // Proceedings of the 3rd ACM Workshop on Scalable Trusted Computing. New York: ACM, 2008: 21 - 30.

[9] Dietrich K, Winter J. Towards customizable, application specific mobile trusted modules[C]//Proceedings of the Fifth ACM Workshop on Scalable Trusted Computing. Chicago: ACM, 2008: 21 - 30.

[10] Fitzek A. Development of an ARM TrustZone aware operating system ANDIX OS [EB/OL]. [2018 - 05 - 09]. https://pure. tugraz. at/ws/portalfiles/portal/1937540/AndixOS_Final. pdf.

[11] Ekberg J E, Kostiainen K, Asokan N. Trusted execution environments on mobile devices[C]//Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security. Bernlin: ACM, 2013: 1497 - 1498.

[12] Santos N, Raj H, Saroiu S, et al. Using ARM TrustZone to build a trusted language runtime for mobile applications[J]. *ACM SIGARCH Computer Architecture News*, 2014, 42(1): 67 - 80.

[13] Linaro. LAS16 - 504: Secure storage updates in OP-TEE [EB/OL]. [2018 - 05 - 18]. http://connect. linaro. org/resource/las16/las16 - 504.

[14] Forissier J. Optee. secure storage[EB/OL]. [2018 - 05 - 07]. https://www. slideshare. net/linaroorg/las16504-secure-storage-updates-in-optee.

[15] Hein D, Winter J, Fitzek A. Secure block device: secure, flexible, and efficient data storage for ARM TrustZone Systems[C] // Trustcom/BigDataSE/ISPA. Washington DC: IEEE, 2015: 222 - 229.

[16] Li X, Hu H, Bai G, et al. Droidvault: a trusted data vault for Android devices[C] // Engineering of Complex Computer Systems (ICECCS). Tianjin: Engineering of Complex Computer Systems(ICECCS), 2014: 29 - 38.

[17] IETF. PKCS #5: Password-based cryptography specification: PBKDF2. [EB/OL]. Version 2. 0. [2018 - 05 - 10]. https://tools. ietf. org/html/rfc2898/#section - 5.

[18] Microsoft. Atomic-writes-in-a-file [EB/OL]. [2018 - 05 - 18]. https://blogs. msdn. microsoftcom/adioltean/2005/12/28/how-to-do-atomic-writes-in-a-file.