Computer Science and Application 计算机科学与应用, 2021, 11(6), 1679-1688
Published Online June 2021 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2021.116173

Hans汉斯

基于双文件系统存储方式的数据安全删除方案

廖麻庭，甘　刚

成都信息工程大学网络空间安全学院，四川 成都
Email: 2502375898@qq.com, test_me@cuit.edu.cn

收稿日期：2021年5月10日；录用日期：2021年6月8日；发布日期：2021年6月15日

摘　要

随着安卓手机的发展，数据删除已经成为人们保护隐私数据的重要手段，但是安卓手机自带的数据删除是不安全的。自带的数据删除是为了回收存储空间所作的一个快速操作，其操作只是将元数据和文件内容的连接阻断，同时把元数据标识成删除标记，但是它清除的文件还是保存在存储介质内，黑客仍然可以通过相应的数据恢复技术将数据还原，从而导致用户的隐私数据泄露。本文就现阶段的数据删除技术进行深度剖析，设计出一种基于双文件系统存储方式的数据安全删除方案。该方案与之前研究人员的安全删除方案进行多项指标对比分析。发现该方案整体优于之前研究人员的方案，方案评估表明该方案符合安全删除的高效性与可行性。

关键词

安卓，隐私数据，安全删除

Data Security Deletion Scheme Based on Dual File System Storage Mode

Xiuting Liao, Gang Gan

School of Cyberspace Security, Chengdu University of Information Technology, Chengdu Sichuan
Email: 2502375898@qq.com, test_me@cuit.edu.cn

Received: May 10th, 2021; accepted: Jun. 8th, 2021; published: Jun. 15th, 2021

Abstract

With the development of Android phones, data deletion has become an important means for

文章引用: 廖麻庭, 甘刚. 基于双文件系统存储方式的数据安全删除方案[J]. 计算机科学与应用, 2021, 11(6): 1679-1688. DOI: 10.12677/csa.2021.116173

people to protect private data, but the data deletion that comes with Android phones is not safe. The built-in data deletion is a quick operation for reclaiming storage space. The operation is only to block the connection between metadata and file content, and at the same time, mark the metadata as a deletion mark, but the files it clears are still stored in the storage medium. Hackers can still restore the data through the corresponding data recovery technology, resulting in the leakage of the user's private data. This article conducts an in-depth analysis of the current data deletion technology, and designs a data security deletion scheme based on the dual-file system storage method. This program compares and analyzes multiple indicators with the previous researcher's secure deletion program. It was found that the program was better than the previous researcher's program as a whole, and the evaluation of the program showed that the program meets the efficiency and feasibility of safe deletion.

Keywords

Android, Private Data, Safely Delete

1. 引言

伴随着智能手机不断的深入发展，其操作系统类型也是百花齐放，例如曾经的 Symbian (塞班)、Web OS、Windows Mobile、Bada、Windows CE 等，然而，发展到今日，智能手机操作系统已经是 Android 和 IOS 的天下了，虽然依然有部分小众的手机操作系统，但是却也动摇不了这两个巨头的地位。截至 2020 年 3 月，我国手机网民规模达到了 8.97 亿，比 2018 年底增加 7992 万人。根据数据调研机构 Kantar 公布的 2018 年第二季度全球各个市场上智能手机所用系统占比情况的统计数据，发现在苹果的大本营美国、日本、英国和澳大利亚市场上，苹果手机所占的比例较高，均超过了 30%，而其中日本的苹果手机比例已经超过了 40%，达到了 42.9%之高，几乎以一己之力打平了全球所有的安卓手机厂商；然而在中国苹果手机的占比却低了许多，只有 19.4%，并且这个数据还在持续下降，可以看出这跟国产手机厂商们的崛起有着不可分割的关系。安卓智能手机给人们生活带来了许多便利，这是毋庸置疑的，但是在给人们带来便利的同时，又获取了使用者大量的隐私数据。人们为了操作更为方便，将许多隐私数据存放在手机中，当手机遗失、转售的时候，隐私数据也就可能会被泄露。安卓智能手机都自带数据删除功能，但是自带的删除功能仅仅是切断了文件元数据与文件内容的链接，并没有将数据从存储介质中擦除，黑客依然可以通过相应数据恢复技术对数据进行恢复操作。因此急需一个有效合理的数据安全删除方案，该方案可以满足对安卓用户数据安全删除的高效性和可行性。本文所研究内容是在双文件系统存储方式下，提出一个合理的数据安全删除方案，以此来实现对数据的安全删除。

2. 安卓数据删除研究现状

针对删除数据可复原的情况，我国研究人员对其做了相应的研究与分析：学者 Tang 等人以加密为依托，总结出一种基于加密的文件安全存储，密钥上传至云端。此工作的假设就是云与用户设备相比，前者的安全性高一些，但是实际上并不是这样。长期以来，终端数据安全这一研究课题得到学者们的重视，不管是在 PC 端，或是在移动终端，两者都是十分关键的，能够使数据自出现到消失的过程的有效性、科学性、实用性等得以保证，这是本文的主要目的。

研究人员 Jia 等人[1]主要研究的对象是 Flash 型安全存储器，主要分析的内容是其中如何将敏感数据进行完全清除，基于他们的研究，最终设计了 NFPS 的擦除方法，在擦除的过程中是通过页面重新定义边界以及部分块擦除的方案来进行操作的。需要注意的是，使用这种方案主要运用在 FTL 层，所以无法应用到较多场景。

研究人员 Chen 等人结合 Flash 设备，将其中的读写单元作为基础进行设计，自 FTL 层入手，使数据擦除方案得以实现，再对 FTL 地址转换机制进行变更，通过数据随机分配表把 FLASH 闪存数据向物理地址进行转化，将内存清洗机制加入进来，使数据得以有效删除。但是，这一方案依然有严重约束的，所以也无法在较多的场景中进行使用，同时如果想要将这种方法运用在终端中也会存在一定的困难。

参考上述研究人员设计的各种安全删除方案可以发现，方案无法在较多的场景中进行使用。所以设计一个能广泛应用的数据安全删除方案是非常重要的。

3. 双文件系统存储方式设计

3.1. 双文件系统存储

双文件系统[2]，其主要是 Android 操作系统中包含两个文件系统。具体的设计方法有多种选择，其中可以通过直接开辟空间法来实现，也可以通过平行双文件系统[3]来进行实现。其中直接开辟空间法具体的实现方法如下，第一步需要找到 Android 终端设备中的 SD 卡后半部分，将这部分的存储空间转化为隐藏空间，下一步逐段初始化 SD 卡，然后可以基于此完成建议模式化的文件系统设计，最终实现隐藏文件的获得，这种方法的缺点在于，其数据存储密集度相对较高。在应用相应的工具检测系统磁盘时，很容易发现异常数据，同时由于数据是按照一定的顺利进行写操作，使得破译数据较容易，因此数据存储的安全性较弱。

下面对于平行双文件系统的方案进行分析，这个方案是在 Android 中包含的 Ext4 文件系统作为基础实现的，在这个结构的基础上，使其在 SD 内部完成平移，这个过程将会经过一个固定的距离，最终得到文件系统。这个文件在系统初始化以后就被隐藏了，通过这种方式可以将一些隐秘的数据在其中进行存储，其不足在于系统初始化的过程中会干扰可见文件中的相关数据。并且通过这种方式进行隐藏，还有可能通过一些磁盘分析工具来对于其结构进行反推。

索引方法主要是临时创新一个索引表，该表的内容是将隐藏文件系统占用的 SD 卡数据块，其与可见文件系统的数据块进行比较。在未覆盖可见文件系统的元数据条件下，隐藏文件系统所占用的整体空间不受约束，尽管物理属性上隐藏文件所占空间呈不连续性，但在系统进行相应的初始化操作后，物理属性上的不连续性将会在逻辑上呈连续状态，从而形成一个新的 linear。当隐藏文件系统进行数据信息读写操作时，将其视为状态下的数据块，其数据格式为 Ext4 模式，与实际的 Ext4 文件系统操作一致。本文所设计的双文件存储方式如图 1 所示。由于该方式生成的文件系统位置呈多变性，因此很难通过磁盘检索工具发现变化规律，在实际应用过程中安全性较高。

为了实现双文件系统存储模式，需要在满足传统单一模式的 Android 操作系统基础上，构建一个新的文件系统，在常规模式下会使该文件系统处于隐藏状态，使得隐藏文件系统包含的数据信息处于隐藏模式，不易被检索。用户可以采取特殊的检索工具获取对应的隐藏文件系统中包含的数据内容，但是其他人很难找到隐藏文件系统的存在，更无法查看隐藏文件系统内包括的数据信息。这样将会在一定程度上显著的增加文件的安全性，使得文件安全性更高。

在设计完成双文件系统的实施策略后，其中又一个难点，也就是其中的两个文件系统之间拥有数据块冲突的情况，之所以会出现这个问题，主要的原因在于这两者都是通过 Ext4 文件系统来构建对应的框架的，还可以发现其中的可见文件系统对于隐藏系统的存在也不了解，所以在分析的过程中选择的是可

见文件系统进行分析，它并没有对于 SD 卡中的组成架构产生任何影响，与写入的磁盘分配策略也完全保持一致，这种情况下就会出现显著的矛盾点：如果可见文件系统内写入来大量文件数据信息，这种情况下存在一定概率覆盖隐藏文件系统数据，这是不符合数据存储规则的。因此，在满足实际条件的基础上调整 Ext4 基础层的数据块分配策略，以使可见文件系统在写入数据时，能够在一定程度上避免占用隐藏文件系统的数据块，从而确保隐藏文件系统的数据存储的完整性。

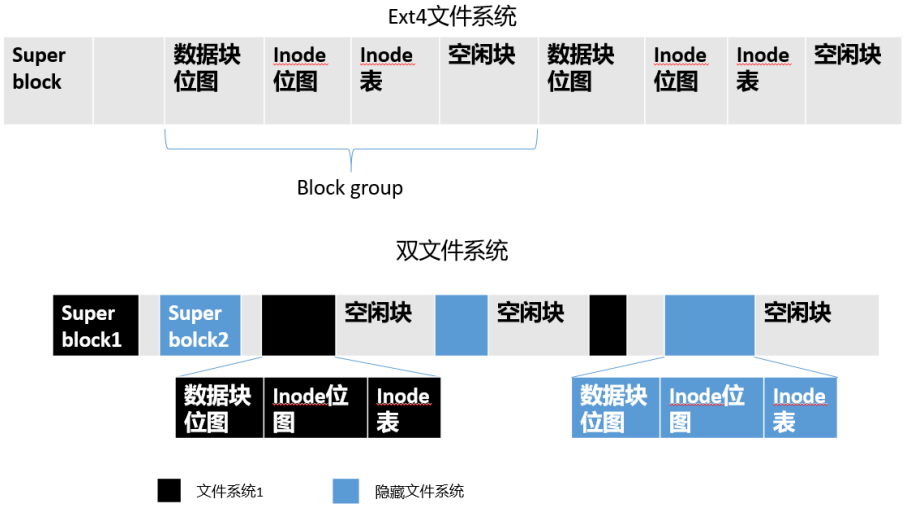

Figure 1. Dual file system
图 1. 双文件系统

3.2. 修改 Ext4 文件系统 Inode 分配策略

作为 Android 操作系统新一代主流文件系统，Ext4 [4]相较 Ext3 具备许多独特的功能。Ext4 文件系统将 SD 卡存储空间划分为不同的空间块组，每个块组将会匹配一个块组描述符，同时将包含相应参数量值的等比例磁盘块。块是文件系统中数据的分配单位，每个块都具有唯一性。通常情况下，在 Android 内部的文件系统中，每个磁盘块具有 4 KB 的存储空间，每个块组包含 32,768 个数据块，即 128 MB 的存储空间。设置 128 M 大小的原因在于，每个块组的块位图均会占用一个磁盘块，而一个磁盘块恰好包含 32,768 个比特。因此，在该模式下，其对应的块组的块位图仅匹配一个磁盘块，在满足实际应用条件的基础上使得后期查找工作更加便捷。基础块分配装置将尝试分配同一块组中每个文件占用的数据块，以降低整个文件磁盘碎片对 I/O 指令的干扰，提高 I/O 应用效率。Ext4 文件系统使用 mode 描述文件相应的数据，每种 mode 均会匹配一个节点号，在实际应用时通过检索节点号将可以查找指定的文件。

数据的局部性对单个擦除块的写入产生影响，可以加速文件重写的速度。因而尽可能减少碎片是必要的，需要对 Ext4 文件系统的底层 Inode 分配策略进行修改，修改内容如下：首先，为了减少磁盘碎片，需要进行多块分配，在文件第一次被创建的时候，会有 8KB 的磁盘空间被块分配器分配给文件，在文件被关闭的时候，将释放所有没有被使用的空间，若该假设为真，那么 extent 中将会存入多个块的文件数据；其次，延迟分配，倘若一个文件需要写入多个数据块，那么新数据在磁盘中的存放位置将会被文件系统推迟决定，一直等到 buffer 写到磁盘为止；然后，尽量保持文件的数据块与改文件数据块的 inode 在同一个块组中，这样可以减少磁盘的寻道时间；接着，尽量保持同一个目录中的所有 inodes 与目录位于同一个块组中；最后，将磁盘卷被分成 128 MB 的块组，在根目录中创建新的目录时，inode 分配器扫描块组并将新目录放到它找到的使用负荷最小的块组中，这可以保证目录在磁盘上的分散性。

4. 数据安全删除方案实现

4.1. 文件系统加密

为了提供足够强度的数据加密，本文采用 AES128 加密方式，AES128 意思是 128 位分组对称加密算法，AES128 本身是一种分组密钥，该算法密钥有 128 位的长度，它的输入数据的长度也是 128 位。AES 集高效率、安全、高性能并且灵活易用等众多优点于一身，它取代了 DES 数据加密算法(56 位密钥)，成为美国的高级加密标准算法，是一种加密强度更高的算法，AES 的 128 位密钥比 DES 的 56 位密钥的加密强度高出 1021 倍之多。

隐藏文件系统中数据的透明加密的设计与实现需要有 cryptsetup 功能文件来直接对文件系统进行加密。cryptsetup 是 Linux 操作系统下的一个磁盘分区加密功能模块，cryptsetup 在指定的磁盘位置进行加密，生成块设备，创建新的文件系统，然后挂载在指定的目录下。cryptsetup 命令加密方式的工作环境非常靠近 Linux 系统的底层，几乎是纯软件加密方式所能达到的最靠近系统底层的。使用 cryptsetup 命令需要下载一个 cryptsetup 功能文件，并将功能模块存放在 Android 文件夹下，赋予其可运行的权限，然后为 cryptsetup 命令设置参数，并运行该命令就可以对指定的 SD 卡分区的文件进行加密，cryptsetup 命令的参数包括需要加密的磁盘块(之前的 dm table 文件)、隐藏文件系统挂载点、需要生成的设备文件名以及自己设置的加密秘钥。加密完成后就会生成相应的设备名，并覆盖之前 dm setup 命令生成的文件系统名，成为加密文件系统。当有新的文件写入时，数据也会进行自动的加密，当使用文件系统卸载命令 umount 或者重启 Android 设备卸载文件系统之后，只有提供相同的密码和 dm 文件才能恢复加密文件系统，这就保证了隐藏文件系统中文件数据的加密安全性。

为了实现应用层可调用的数据透明加密服务，需要在 Android 应用框架层编写相应的 System Service 来调用 cryptsetup 功能文件。cryptsetup 功能文件是已经存在的模块，将其放在/data 目录下，并赋予其可执行权限，接下来需要编写应用框架层的透明加密 System Service 来调用 cryptsetup 功能文件，实现数据透明加密服务，最后在应用层开发 App 来调用透明加密服务，实现并验证数据透明加密的功能。

为了实现应用层可调用的数据透明加密服务，需要在 Android 应用框架层编写相应的 System Service 来调用 cryptsetup 功能文件。cryptsetup 功能文件是已经存在的模块，将其放在//data 目录下，并赋予其可执行权限，接下来需要编写应用框架层的透明加密 System Service 来调用 cryptsetup 功能文件，实现数据透明加密服务，最后在应用层开发 App 来调用透明加密服务，实现并验证数据透明加密的功能。

本文提出的数据透明加密系统服务在基本框架分为四个层次，自上而下分别为 SDK 接口层、IPC 接口描述层等[5]，图 2 为 System Service 框架图，详细介绍如下：

SDK 接口层：SDK 接口层是安全接口可以扩展的关键。通过 System Manager 模块对相应的系统接口进行封装，App 可以调用 getService()方法[6]获取相应的 System Manager 类，同时 Context 对系统服务和自定义服务进行统一的命名，将该命名作为 getService()的参数传递给 lPC 接口层来实现功能调用。

SDK 接口的设计思路如下：

1) 在 Android 应用框架层添加自定义的 System Service，获得加密服务。

2) 在 Service manager 文件中注册 System Service，以保证 System Service 在 Android 操作系统启动后能够自动运行该服务。

3) 将自定义的 System Service 通过定制 IPC 接口描述层和 SDK 接口层实现外界调用接口的开发。

IPC 接口描述层：IPC 接口描述层调用 SystemService 接口并向 SDK 提供服务。在 IPC 接口层中，aidl 文件是一种接口描述文件，ContextImpl 文件提供 Context 参数，这两个文件是 Binder 机制的关键。

System Service 功能实现层：数据透明加密服务最重要的功能函数就是在这一层实现的，System

Service 通过调用系统内核的函数和接口以及系统本身具有的其他 System Service 来实现自己的服务功能，实现功能扩展。这一层也是系统服务可扩展的关键。在 Android 应用框架层，System Service 和 System manager 交互，在 Service 列表中注册，保证系统服务的正常启动并在系统后台运行。System server 统一的注册、管理和启动所有的 System Service。所以，不管是开发者自定义的 System Service 还是 Android 自身的 System Service，都必须把自身相应的信息注册在 System Server 之中，并通过类似于 ServiceManager.addService (ComtExt.OPERSYS_SERVICE)的方法来获取 System Service 的功能。System Server 会向 System Manager 中提供自己的服务信息，System Manager 中的函数接口可以被 App 或者其他的 System Service 调用，从而真正实现服务的调用。System Service 既可以直接调用 Android 应用框架层的其他 System Service 也可以通过 jni 的方式调用 Android kernel 层中 C/C 一提供的系统调用。Binder 实现 Service 之间的通信功能，App 或者 System Service 在调用其它 System Service 时，需要通过 Binder 机制实现。实现 Binder 机制需要定义一个 aidl 文件[7]，aidl 文件是一个格式要求严格的接口文件，其内容是供外界调用的一系列接口函数。

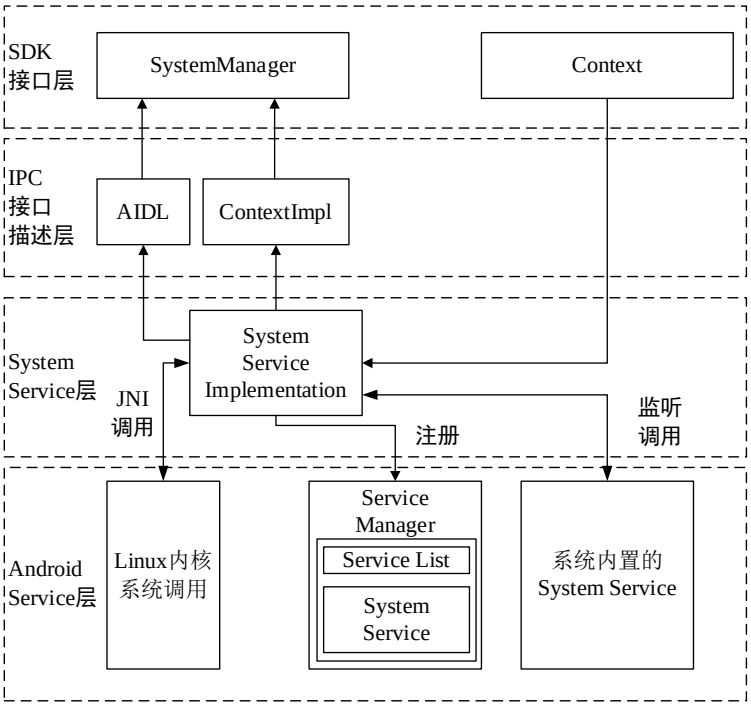

Figure 2. System Service framework diagram
图 2. System Service 框架图

在 Context 文件中添加上述服务后，App 或者其他的 System Service 就可以通过方法 IOpersysService.Stub.asInterface(ServiceManager.getService("opersys"))来调用该 System Serviceo。

Android Service 层：Android Service 层包含所有的 Android 系统服务以及 Android 向应用层提供的接口。本文将通过封装和调用这些功能接口来实现数据透明加密服务。

4.2. 密钥管理模块

为了提升系统安全性，保障系统稳定运行。密钥管理模块应运而生。密钥存储区是这一模块管理的重点。存储区密钥生成主要由用户口令及 TrustZone [8]所内置的 Key 实现，其需要达到的功能为有效管

理文件系统配置文件及相关密钥，全部加密文件密钥都会被集中在存储区管理。密钥管理模块负责存储区的管理。删除文件密钥就可以删除文件，确保操作更加安全。还有就是密钥管理模块可以提供紧急删除文件系统这一选择。对于预置校验口令进行对比判断，可以有效删除密钥存储区，实现对于废弃闪存页的回收。

4.3. 安全删除开销优化

由于存储特性所决定，存储块状态主要可以分为无效、有效、空闲这些类型。在具体应用当中如果存储介质面是 N 个块形成，这里面的每个具体块由 M 个页构成，由此就可以实现把整个存储介质单元阵列抽象为 $N\times M$ 二维矩阵，页作为这里面最小单位。这就可以使用到 $State_{[i][j]}\left(1\le i\le N,1\le j\le M\right)$ 使得表示页各种状态得到体现，i 是该矩阵里面第 i 列(第 i 个物理块)，j 是矩阵里面第 j 行(第 j 个页)，具体状态为：

$State_{[i][j]}=1$：属于无效页，数据无效，将被安全删除无效页；

$State_{[i][j]}=0$：属于有效页，数据有效，要保证数据安全性，避免被删除，保障用户有效访问；

$State_{[i][j]}=-1$：属于空闲页，没有存储数据，不存在约束。

本次研究提出的方案目的在于能够有效删除无效数据，具体方法结合到块擦除和密钥删除等方法，确保无效数据被安全删除。如无效数据出现后，可以借助于密钥删除进行数据删除操作。用 $Dblock_{[i]}$ 标记第 i 个物理块是不是启动块擦除操作，当 $Dblock_{[i]}=1\left(1\le i\le n\right)$，表示第 i 物理块会启动擦除，清除数据；当 $Dblock_{[i]}=0$，表示第 i 物理块不需要启动擦除，清除数据。同样的道理，用 $Dkey_{[i][j]}$ 标记第 i 个物理块里面第 j 页，对称密钥是不是发起删除操作。当 $Dkey_{[i][j]}=1\left(1\le j\le m\right)$ 时，第 i 个物理块当中第 j 页就会启动密钥删除的具体操作；当 $Dkey_{[i][j]}=0$ 第 i 物理块第 j 页不会启动密钥删除具体操作。

为了能够确保删除操作均符合安全删除操作要求可以实现对于无效数据的清除，确保有效数据依然可以正常访问。安全删除这类问题能够运用数学模型形式化进行具体描述：

$$\forall i,j\left(1\le i\le n,1\le i\le m\right)$$

$$Dblock_{[i]}+Dkey_{[i][i]}\ge State_{[i][i]}$$

且当仅当：

$$State_{[i][j]}=0,Dkey_{[i][j]}\ne 1$$

上面式子里面 $State_{[i][j]}=1$ 时，$State_{[i][j]},Dkey_{[i][j]}1$ 里面有一个应该是 1，代表能够实现无效数据删除操作。那么 $State_{[i][j]}=0$ 时，就代表这一页所属于的物理块有其他无效页，想要在这一区域实现块擦除，则需要将数据进行转移。当 $State_{[i][j]}=-1$ 时，就表明这一页为无数据页，相应删除操作就属于无效操作。$State_{[i][j]},Dkey_{[i][j]}1$ 满足式子要求。在达成条件背景下，进一步优化安全删除操作行为。

5. 实验分析及评估

5.1. 实验内容

利用实验对密钥派生过程中所涉及的时间、数据解密及加密的时间、不同数据量写入与读取的时间以及操作删除数据期间的安全验证方案的可能性进行分析。本次实验中所使用的设备配置有：计算机处理器(CPU)英特尔酷睿 I5-4539、CPU 单核频率 3.30 GHz；计算机内存(RAM) 8 G；计算机硬盘 1024 GB；计算机操作系统 Ubuntu14.04；固态硬盘及模拟器为 SSDModel 和 DiskSim4.0。实验初期将负载踪迹文件

与结构文件输入到虚拟硬盘中，然后将虚拟硬盘的储存空间进行划分，划分设定为 8 组，每个组包含 8 个面，每个面包含 4096 块，每个块包含有 64 页，每页的数据量为 4 KB；虚拟硬盘中的 8 组中均分别具有独立的主密钥，因各组均采用单一专属的时间处理单元，且具有独立性。通过负载踪迹文件来记录负载请求的时间、类型等信息。通过输出的文件数据信息的记录请求响应时间与读写请求信息时间以及时间开销等数据加密选择使用 AES-256 对称加密算法，该密钥长度设置为 256 bits，此次试验的密钥加密方案中所涉及的哈希算法则为 SHA-256 算法。在具体的试验过程中，首先进行虚拟密钥测试，根据存储器的存储结构及性质进行设置，并设置为 1~6 层，并测试不同层级中节点密钥的具体运行时间。最终的试验结果如图 3 所示，本次所提出的试验方案是有效的，随着层级的上升，时间开销逐渐增长，但时间开销均保持在客户端可接受范围以内。

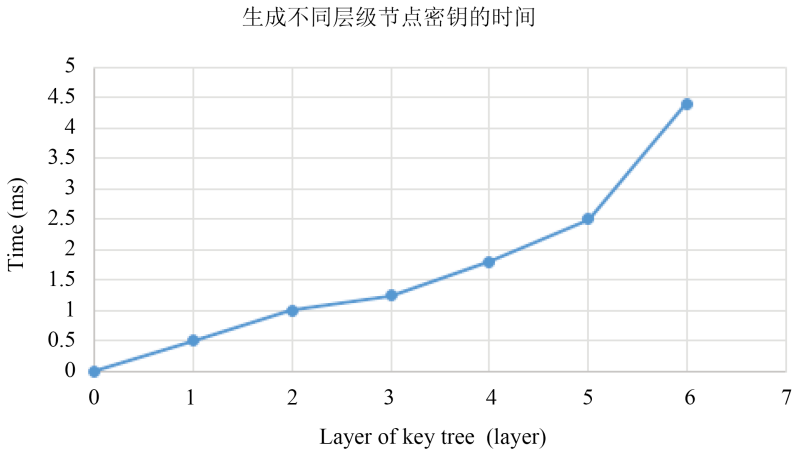

Figure 3. Time to generate keys for different levels of nodes
图 3. 生成不同层级节点密钥的时间

接下来在 MSR-Cambridge 中筛选出 8 MB 至 512 MB 不同容量的数据进行试验，通过试验来测试加密与解密时间，本次试验中所选取的具体数据块的数据量分别为：8 MB、16 MB、32 MB、64 MB、128 MB、256 MB、512 MB。实验中对每组数据均循环执行 100 次，然后取平均值作为最终的试验结果。试验中测试数据的加密(Encrypt)与解密(Decrypt)时间开销的对比结果如图 4 可见，随着数据块数据量的增大，其所进行的解密与加密的时间开销也在不断地增长，并呈逐渐增长的趋势。

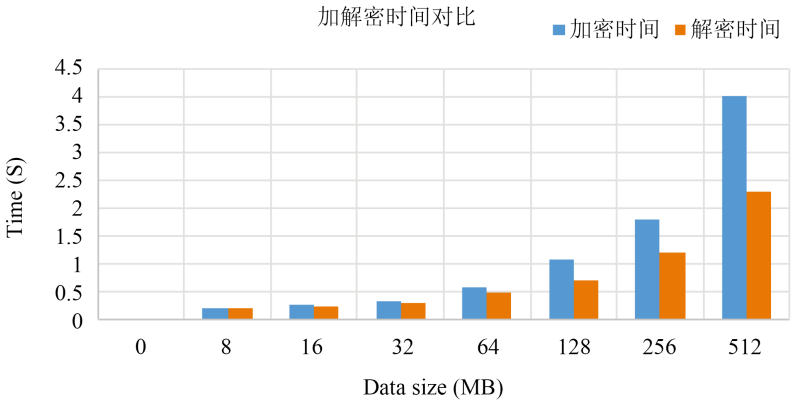

Figure 4. Data encryption and decryption time comparison
图 4. 数据加密和解密的时间对比

如图 5 内容所示，在测试文件模块中不同数据量的读写数据的速度比中，当读写小于 64 KB 时，读写的速度会因为读写模块数据量的增长而提高速度，但如果超过 64 KB 时，则或呈明显减弱的现象。

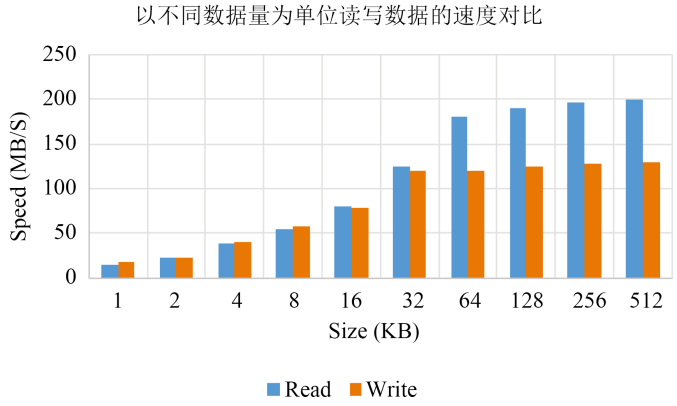

Figure 5. Comparison of the speed of reading and writing data based on different data volumes
图 5. 以不同数据量为单位读写数据的速度对比

与其他研究人员所提方案进行对比：

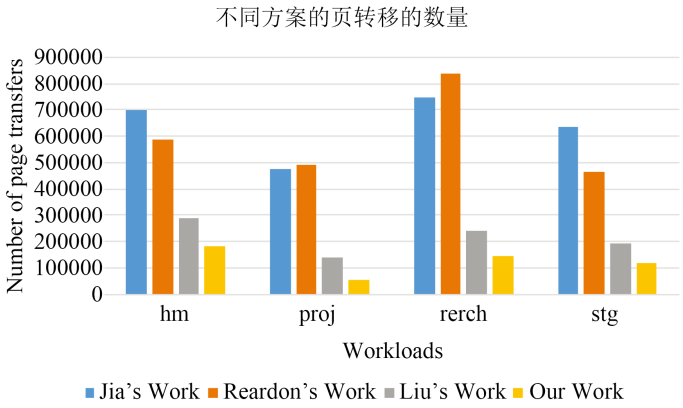

Figure 6. The number of page transfers in different scenarios
图 6. 不同方案的页转移的数量

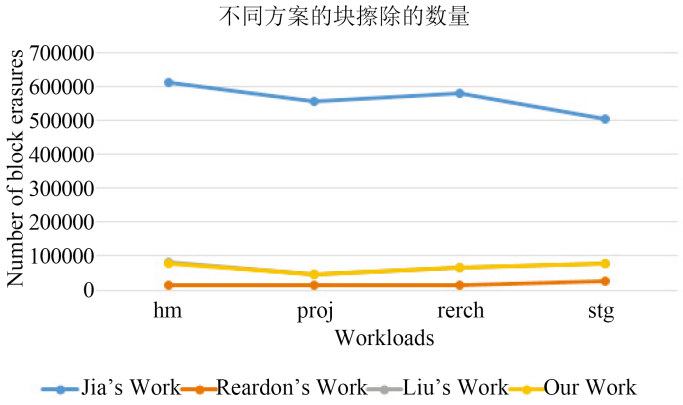

Figure 7. The number of erase blocks in different schemes
图 7. 不同方案的擦除块的数量

5.2. 实验结果分析

最终的试验结果也表明，本次所提出的试验方案是有效的，随着层级的上升，时间开销逐渐增长，但时间开销均保持在客户端可接受范围以内。通过本次试验可发现，安全删除的开销主要以页的转移量与块的擦除数来进行反映，试验选择 MSRCambridgetraces 数据作为负载，具体选择了 hm、proj、rerch 及 stg 四种负载，分别在不同的时间内进行操作，具体为：54、85、119 及 76 小时内每秒中请求 14 次数据操作。如图 6 可见安全删除测试过程中的不同方案的页转移数量，图 7 可见不同方案的块擦除的数量。本文中所提出的安全删除方案中，因密钥派生加密来实现数据加密，因此在密钥删除中仅删除哈希值便块擦除数。Reardon 及 Liu 等人[9]所提出的方案中则是采用了密钥加密多份数据，因此在进行无效数据删除的同时也将有效数据进行了转移。综上，本文设计方案中的页转移次数小于 Reardon 等人和 Liu 等人所提出的方案，我们所提出的方案整体优于之前研究人员的方案，方案评估表明该方案符合安全删除的可行性与高效性。

致　谢

在此感谢我的导师甘刚教授。虽然甘老师很少在学校，但是依然十分认真负责地指导我的论文，为我的论文提供了很多建设性的意见，这篇论文不论是选题、创新还是定稿，甘老师都十分耐心地为我提供建议与解答各种疑惑。甘老师的专业知识非常渊博，对于学术也十分严谨，是我学习和奋斗的榜样。

参考文献

[1] Jia, S.J., Xia, L.N., Chen, B., *et al.* (2016) NFPS: Adding Undetectable Secure Deletion to Flash Translation Layer. 11*th ACM Conference on Asia Computer and Communicatioins Security*, Xi'an, May 2016, 305-315. https://doi.org/10.1145/2897845.2897882

[2] 丁云冰, 杨戈, 卜凡. Android 智能终端中基于日志的文件恢复方法[J]. 长春工程学院学报(自然科学版), 2019, 20(3): 81-84.

[3] 曹震寰, 蔡小孩, 顾梦鹤, 顾小卓, 李晓伟. 基于访问控制列表机制的 Android 权限管控方案[J]. 计算机应用, 2019, 39(11): 3316-3322.

[4] Mathur, A., Cao, M., Bhattacharya, S., *et al.* (2007) The New ext4 Filesystem: Current Status and Future Plans.

[5] Meier, R. (2008) Professional Android Application Development. Wrox Press Ltd., Birmingham.

[6] 怡凯. 基于 Android 的移动空间信息服务研究[D]: [硕士学位论文]. 大连: 辽宁师范大学, 2009.

[7] 王鑫. Android AIDL 应用技术研究[J]. 信息技术, 2015(7): 193-194.

[8] 韦腾. 面向移动终端的数据安全删除方法研究与实现[D]: [硕士学位论文]. 西安: 西安电子科技大学, 2018.

[9] Liu, C., Khouzani, A.H. and Yang, C. (2017) ErasuCrypto: A Light-Weight Secure Data Deletion Scheme for Solid State Drives. *Proceedings on Privacy Enhancing Technologies*, **2017**, 132-148. https://doi.org/10.1515/popets-2017-0009