# 设计和实现基于 UsbKey 的透明加解密文件系统

刘威鹏[1] 胡 俊[1] 刘 毅[2]
(中国科学院研究生院信息安全国家重点实验室 北京 100039)[1]
(解放军信息工程大学电子技术学院 郑州 450052)[2]

**摘 要** 加密文件系统是在操作系统级保护系统和防止用户敏感数据泄漏的一种有效手段。首先对几个著名的加密文件系统地进行了分析,并指出其在使用方式、密钥保护、配置管理以及性能方面存在的问题,而后从设计目标、系统组成、实现三个方面分析和讨论了一种基于 UsbKey 的透明加解密文件系统的实现方案,该方案基于 Linux 操作系统,使用 UsbKey 对于用户密钥进行安全存储和保护,并利用 LSM(Linux Secure Module)机制,通过对于 inode 节点的读、写等关键操作的重定向,来实现文件的透明加解密的功能。本文对于 Linux 操作系统下的透明加解密文件系统的设计与实现具有一定的指导意义。

**关键词** UsbKey,透明加解密,文件系统,Linux 安全模块

## Design and Implementation of Transparent Encrypted and Decrypted File System Based on UsbKey

LIU Wei-peng[1] HU Jun[1] LIU Yi[2]
(State Key Laboratory of Information Security of GSUAS, Beijing 100039, China)[1]
(Electoronic Technology Institute of PLA Information Engineering University, Zhengzhou 450052, China)[2]

**Abstract** Encrypted file system is an effective method to protect sensitive data of system and user from disclosing on operating system level. This paper first investigates several famous encrypted file system and points out their limits, such as use method, key protection, management and performance, then discusses a scheme of transparent encrypted and decrypted file system based on UsbKey from design objective, architecture, and implementation. This scheme is based on Linux operating system, at the same time, uses UsbKey to storage and protect key. Using LSM framework and redirecting the "read" and "write" operation of inode implement transparent encrypted and decrypted function. This paper gives some lights on the research and implementation of transparent encrypted and decrypted file system in Linux.

**Keywords** UsbKey, Transparent encrypted and decrypted, File system, LSM

## 1 引言

当前,安全操作系统主要是从认证、授权和审计这三个方面入手,通过检查用户身份、限制用户权限、监视用户行为来保证系统的安全性。通常,这些安全机制能够很好地保护数据的机密性和完整性,防止系统安全性遭到破坏。然而,有时这些安全机制却无能为力,例如攻击者可以通过运行另一个操作系统以管理员权限读取该系统磁盘上的敏感数据。如果通过使用密码技术对敏感数据进行加密,那么攻击者即使能够获取这些敏感数据,但由于没有正确的密钥,也无法读取机密信息,从而能够很好地保护其安全。由于加密文件系统可以提供对数据的加解密服务,因此在安全操作系统中引入加密文件系统的功能,对于系统整体安全性的提高具有重要的意义和作用。但是,由于加密文件系统使用了密码机制,同时会带来密钥分配和密钥存储等多方面安全问题,从而使得人们对于加密文件系统的使用"望而却步"。

UsbKey 是一个集身份认证、密钥安全存储、对称钥和非对称钥加密等安全功能于一体的安全硬件。安全操作系统启动后通过插入 UsbKey 对于用户的身份进行认证;通过把密钥安全存储在 UsbKey 中,能够防止泄漏、篡改并保证密钥不落地;通过管理平台进行的统一的密钥分配和管理,可以在一定程度上消除密钥管理中潜在的安全隐患。因此,如何利用 UsbKey 所具有的上述功能来构建一个透明加解密文件系统就成为当前研究的热点和难点。而本文针对这一研究方向的有益探讨,提出了一个基于 UsbKey 的透明加解密文件系统。

文章组织如下:第 2 节对几个著名的加密文件系统进行分析,并指出它们存在的问题;第 3 节提出一个基于 UsbKey 的透明加解密文件系统(TEDFSU, Transparent Encrypted and Decrypted File System Based on UsbKey)的设计方案,并从设计目标、系统组成、实现以三个方面对 TEDFSU 进行讨论;第 4 节把 TEDFSU 和其他加密文件系统进行比较;最后总结全文,并指出了今后研究和改进的方向。

## 2 相关工作

### 2.1 CFS(Cryptographic File System)

CFS 是一个广泛使用的以用户层 NFS(Network File System)服务器方式实现的加密文件系统[1]。它需要用户在本地或远程文件系统中创建一个目录来保存加密数据,并确定所使用的密码和密钥。CFS 守护进程在验证了用户身份和密钥之后,通过 attach 命令使用户可以透明访问自己的加密

刘威鹏 博士研究生,研究方向为安全操作系统和可信计算;胡 俊 博士研究生,研究方向为安全操作系统和可信计算;刘 毅 博士研究生,研究方向为安全操作系统和可信计算。

目录中的文件。CFS对用户来说并不是完全透明，用户对加密目录进行操作之前，需要用命令先将加密目录和一个特定目录相关联；加密的粒度是目录，不能对特定的文件加密；效率比较低，由于运行在用户模式，必须执行在用户空间和内核空间的上下文转换和拷贝。

### 2.2 TCFS(Transparent Cryptographic File System)

TCFS是内核级的NFS客户机，使用时需要和一个NFS服务器关联[2]。和CFS相比，TCFS加密的粒度较细，不仅能够对每个文件使用不同的“文件密钥”进行加密，而且能够对文件中不同部分使用的不同的“块密钥”进行加密。TCFS对数据加解密操作在核心层完成，性能有所提高，但是用户的“主密钥”由用户的登录密码加密后存放在系统硬盘中，这在一定程度上降低了系统的安全性。此外，TCFS目前只在Linux 2.2.17之前的版本上实现，兼容性差。

### 2.3 CryptFS(Crypt File System)

CryptFS是一个分层的虚拟节点(vnode)接口[3]。Linux中和虚拟节点相对应的是索引节点(inode)。虚拟节点最重要的方面是对文件分层操作，虚拟节点位于最高层，提供虚拟节点接口。当高层的操作系统代码调用虚拟节点接口进行文件操作的时候，虚拟节点会调用和具体的文件系统相关的操作完成具体的文件操作。通过上层的虚拟节点调用下层的节点，下层的节点调用次下层的节点，最后就能完成文件操作。CryptFS在虚拟节点层和下层节点层之间加入一个CryptFS层，由CryptFS层自动实施安全操作以后调用下层的节点层(例如调用UFS层)完成操作。CryptFS使用密码块链(cipher block chaining)加密模式，但仅仅提供了Blowfish算法，扩展性差。

### 2.4 AFS(Andrew File Systerm)

AFS是一个分布式加密文件系统[4]，它通过一个统一的访问接口把多个服务器连接起来，形成一个数据存储空间。客户端通过把共享目录挂载到本地目录来访问服务器上数据。客户端和其他客户端以及服务器之间的信息交换通过安全的RPC来完成，每次用户登录都要和服务器协商一个会话密钥，从而每次的消息交换都是通过会话密钥加密进行的。在AFS中，服务器使用明文方式保存数据，因此攻破AFS系统的任何一个服务器都能得到文件的内容。此外，由于使用端到端的加密机制，系统效率受到很大影响。

### 2.5 存在的问题分析

通过分析，以上的加密文件系统都存在着不同程度的问题：

(1)密钥保护不完善。在CFS，TCFS和CryptFS中，由于加解密密钥在硬盘上存储，从而存在受到攻击的危险。

(2)使用配置不方便。在CFS中，用户对加密目录进行操作之前，需要用命令先将加密目录和一个特定目录相关联，并确定所要使用的密钥和密码，而通过图形管理界面，则配置起来比较方便和灵活，易用性好。

(3)针对分布式环境。例如，CFS，TCFS和AFS都是针对分布式环境的，使用时需要采用C/S模式，并不支持单机版的使用。

(4)性能比较低下。在CFS，由于加解密在应用层完成，性能比较低下；而AFS使用端到端的加密机制，系统效率受到很大影响。

## 3 TEDFSU的设计与实现

### 3.1 设计目标和基本原则

借鉴已有的透明加解密文件系统的设计，并根据实际的系统使用环境和安全威胁，确定了TEDFSU的设计目标和原则。

(1)通用性：尽可能地使用Linux已有的系统调用接口，并支持底层文件系统所支持的任何访问方法；

(2)易用性：便于使用，通过图形化管理工具对于用户更易接受；

(3)高性能：系统不应该有太多的性能损失和进程上下文的切换，将透明加解密机制放到内核实现是很好的选择；

(4)安全性：密钥和加密算法存放在安全硬件中比存放在硬盘中安全性好；能够对于登陆系统的用户进行身份认证，采用双因子认证更为安全；

(5)可扩展性：系统采用结构化设计，便于功能的扩展及对多种密码算法的支持。

### 3.2 系统的组成及工作流程

系统包括两个部分：配置管理工具和完成透明加解密机制的安全内核模块。配管工具是使用QT(Linux下图形界面工具)编写的应用程序，主要完成UsbKey发行以及对系统中用户及其所属资源的配置管理工作。配管工具可以发行两种类型UsbKey(可扩展)，一种是0级Key，即特权的管理员Key；一种是1级Key，即普通的用户Key。此外，为便于管理和使用，对UsbKey进行分组，同一组中的用户具有相同的工作密钥，但其私有密钥不同。用户管理包括：用户的添加(把用户名和特定的UsbKey绑定)、删除和修改；用户所属资源的管理包括：对用户所属资源(文件和目录)添加、删除或修改加密属性。

UsbKey中存储的主要信息包括：UsbKey名称，用户登录UsbKey的密码及加密算法，以及公共密钥、私有密钥。和公共密钥、私有密钥相对应的是系统中的公共加密目录和私有加密目录。公共加密目录是使用UsbKey中的公共密钥对其进行加密(文件是以密文的方式在硬盘中存储的)，同一工作组中的所有用户都能够进行透明的读写；而私有加密目录中的文件(通常在/home/下，名称为用户名)，只有属主用户(具有匹配的私有密钥)才能够进行透明的读写，其他任何用户(同组或非同组)所看到的都是密文。

系统的关键部分是可动态加载的透明加解密模块，它是以Linux模块的形式实现，具有灵活性和通用性好的特点。在具体实现透明加解密模块时，使用了Linux的LSM机制，它是Linux下一种灵活的和通用的访问控制框架，同时是在Linux下开发安全功能的标准机制。

系统工作的流程如下：通过对系统的grub和initrd进行修改，使得系统在正常启动过程中自动加载UsbKey的驱动程序模块和用户身份认证驱动程序模块(在UsbKey驱动程序基础上的模块)，而后插入透明加解密安全模块，在对用户的身份成功完成验证后，进入安全操作系统(运行安全内核)，否则退出系统。

配管工具采用向导式的界面。运行前插入管理员UsbKey，并正确地输入UsbKey登录密码后，程序向后执行。添加、删除和修改用户都首先需要插入用户的UsbKey，并且在成功地验证用户的登录密码后才能执行下一步操作。当对资源设置加密属性后，系统会使用UsbKey中的密码和密钥进行加密。对于用户及用户资源的配置管理所产生的最终结果都记录在两个文件中，即os210_userlist和os210_reslist中，这个文件也是安全管理工具和安全内核之间的接口。安全内

核在初始化的时候，读取 os210_userlist 对用户初始化，形成一个系统用户链表，作为后面对用户进行鉴别和认证的基础；同时读取 os210_reslist 文件，形成一个受控资源链表。在系统运行中，当用户读取加密文件后，首先在受控文件链表中进行查询，如果是自己受控的，那么就通过从 UsbKey 中的读取公共密钥或私有密钥对文件进行解密。这些都是利用操作系统的底层的机制，并且通过对 Linux 文件系统的关键系统调用进行修改和重定向来完成的，对上层用户来说是完全透明的，和在普通文件系统下对于文件操作没有什么不同，因此称之为透明加解密文件系统。

### 3.3 实现

Linux 最初采用 minix 的文件系统，在改进并吸取了传统 Unix 文件系统的经验后，形成了现在的 Linux 文件系统：ext2 和 ext3。为了更好地支持各种文件系统，需要把对其操作和管理纳入到统一的框架中，让内核中文件系统界面成为一条文件系统“总线”，使得用户程序可以通过同一个文件系统操作界面，即同一组系统调用，对各种不同的文件系统进行操作。这样就可以对用户程序隐藏各种文件系统的实现细节，从而为用户提供一个统一的、抽象的、虚拟的文件系统界面，这就是“虚拟文件系统 ”(VFS, Virtual File System)。这个抽象的界面主要由一组标准的、抽象的文件操作构成，以系统调用的形式提供给用户程序，如 read()，write()等等。

在 Linux 系统中，只有在“打开”(通过系统调用 open())文件后，才能通过系统调用 read()和 wtite()对文件进行“读”或“写”操作。为提高效率，对文件的“读”或“写”都是带有缓存的。通过缓存机制，系统为最近刚“读”或“写”过的文件内容在内核中保留一份副本，以便再次需要已经在缓存中存储的副本的内容时就不必再从磁盘中读入，而需要写的时候则可以先写入副本中，空闲时再从副本写入磁盘。Linux 中，将装有同一个文件内容的缓冲区都放在其所属文件的 inode 结构中。在 inode 结构中设置了一个指针 i_mapping，它指向一个 address_space 结构，缓冲区队列就在 address_space 中，并以内存页面的形式挂载。在 address_space 结构中，队列头 pages 就是用来维护缓冲页面队列的。指针 a_ops 指向一个 address_space_operations 数据结构，该结构的函数指针给出了缓冲页面与具体文件系统的设备层之间的关系和操作，例如从具体文件系统的设备上“读”或“写”一个缓冲页面等等。对于具体的文件系统 ext2 来说，这个数据结构为 ext2_aops，如下：

```
struct address_space_operations ext2_aops= {
readpage:ext2_readpage;
readpages:ext2_readpages;
prepare_write:ext2_prepare_write;
commit_write:generic_commit_write;}
```

其中，readpage 的功能是从缓冲页面中读单页面操作；readpages 的功能是从缓冲页面中读取多个页面操作；prepare_write 的功能是为一个给定的缓冲区页面中的记录块做好写入准备；commit_write 则通过 prepare_write 为写操作做好准备后，就从缓冲区到设备上的记录之间建立了一条通路。写入缓冲区以后，还要把这些缓冲页面交给内核线程 kflushd，而将缓冲页面提交给 kflushd 进程的任务是由 commit_write 函数来完成的。

为实现文件的透明加解密的功能，我们对上述结构中函数功能进行了重定向，即在系统实现时，对于具有加密属性的文件，在使用 readpage 或 readpages 进行读文件时，要进行解密操作；在使用 prepare_write()和 commit_write()进行写操作时，要进行加密操作。为了更好地实现上述功能，在编码时，编写了专门的重定向函数 os210_cip_inode_redirectop()，该函数包含所有的重定向功能。

```
os210_cip_inode_readirectop{
  new_a_ops->readpage=os210_cip_readpage;
  new_a_ops->readpages=os210_cip_readpages;
  new_a_ops->prepare_write=os210_cip_prepare_write;
  new_a_ops->commit_write=os210_cip_commit_write;
  }
```

整个结构的修改如图 1 所示。

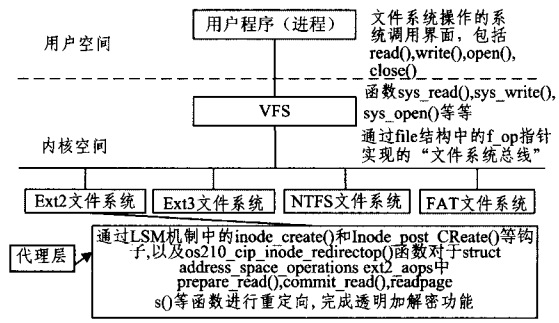


图 1 修改后的 ext2 文件系统

为了便于透明加解密功能的实现，在系统中使用了 LSM 机制。LSM(Linux Security Module)是 Linux 安全模块，它是 Linux 下一个灵活的、通用的访问控制框架。LSM 的基本设计思想是在内核数据结构中放置了透明安全域(Opaque security field)作为内核对象的安全属性，并在内核源代码中放置钩子 hook 函数，由 hook 函数来对内核对象的访问权限做出判断。在实现透明加解密机制时，需要添加安全属性的内核对象为：inode(代表文件)、super_block(代表文件系统)。为了对文件的加密属性进行设置以及读写操作的重定向的工作，主要使用了 LSM 框架的 inode_permission()(该钩子在打开一个文件前检查权限)、file_permission()(该钩子在对于一个文件进行真正的读写操作前检查权限)、inode_post_create()等钩子(该钩子设置一个新创建的普通文件的安全属性，其在文件被成功创建后调用)。通过在这些钩子函数中增加节点加密属性设置函数 os210_set_inode_sec_attr()(其功能是根据索引节点及索引节点的目录节点的安全信息，设置索引节点的安全属性)以及读写操作的重新定向功能 os210_ cip_inode_redirectop()函数(其功能是当文件带有加密属性时，重定向 inode 操作的一些函数，使其指向带有加解密功能的函数)来最终完成透明加解密功能。

在实现时，还根据实际的需要，增添了一个 LSM 框架的定义外的钩子函数：mpage_read()；os210_mpage_read()，并且对内核的文件系统部分的 mpage. c 中的 read_page()函数进行了修改，增加了对该钩子的调用功能，其功能是在执行 readpage 操作时，对读出的页面进行解密操作。

此外，在系统实现时，采用了结构化的设计方法，一方面便于系统的实现(编程)以及具有更好的稳定性和兼容性；另一方面便于对于多种加密算法的支持，使得在系统具有更好的扩展性。

## 4 比较和分析

本节从工作原理、实现位置、数据存储、密钥存放、可扩展

性、使用简易程度、性能等几个方面把 TEDFSU 和其他加密文件系统进行比较，结果见图 2。

| 内容 | CFS | TCFS | CryptFS | AFS | TEDFSU |
|---|---|---|---|---|---|
| 工作原理 | NFS 服务器 | NFS 客户机 | 虚拟节点 | 普通文件系统 | 虚拟节点 |
| 实现位置 | 应用层 | 内核层 | 内核层 | 内核层 | 内核层 |
| 数据存储 | 密文 | 密文 | 密文 | 明文 | 密文 |
| 密钥存放 | 磁盘 | 磁盘 | 磁盘 | 磁盘 | Usbkey |
| 扩展性 | 较差 | 较差 | 较好 | 较差 | 较好 |
| 使用简易 | 较复杂 | 教复杂 | 较复杂 | 较复杂 | 容易 |
| 性能评测 | 较差 | 较好 | 较好 | 较差 | 较好 |

图 2　TEDFSU 和其他加密文件系统的比较

工作原理：CFS 和 TCFS 基于 NFS 文件系统，CryptFs 是基于虚拟节点的工作机制，AFS 基于普通的文件系统的工作机制；TEDFSU 从本质上来讲也是基于虚拟节点的，因为 i-node就是虚拟节点的中重要的一种。

实现位置：通常，在核心层进行加密操作将会得到更高的性能。

数据存储：数据在磁盘中以密文的方式存储比采用明文方式存储显然具有更高的安全性。

密钥存放：和把密钥存放在磁盘上相比，TEDFSU 把密钥存放在 UsbKey 中，具有较高的安全性能。

扩展性：由于 TEDFSU 对内核的依赖性较小，因此便于扩展；由于采用结构化的设计和编程，能够对多种加密算法提供支持。

使用简易：TEDFSU 由于实现了图形化的配管工具，使用简单灵活。

性能评测：CFS 在应用层加密使得在对本地数据进行加密时性能非常低。相比之下，CryptFS 和 TEDFSU 则高效得多。此外，TEDFSU 由于采用硬件加密，和其他采用软算法加密的系统相比，具有更高的性能。

**结束语**　本文设计和实现了一个基于 UsbKey 的透明文件加解密系统，并且从设计目标、系统组成和工作流程以及实现 3 个方面对其进行了讨论，最后把它和其他加密文件系统做了比较。和传统的加密文件系统相比，TEDFSU 的密钥和加密算法安全存储在 UsbKey 中，只有合法的用户拥有 UsbKey 并且输入合法的口令后，才能够登录系统。并通过实现一个图形管理界面解决了常见加密文件系统易用性较差的问题。此外，在系统中文件以密文的方式存储在磁盘中，只有合法的用户查看文件内容是明文，并没有感觉到加密机制的存在，而非法的用户得到的却是密文，实现了透明加解密的功能。

该系统经过测试证明已经能够很好地支持 Linux 文件系统 ext2 和 ext3。下一步工作之一是 Windows 的 NTFS 和 VFAT 等文件系统进行移植的工作。此外，由于该系统采用 LSM 机制和结构化设计，扩展性和灵活性较好，因此我们还将研究如何把对于文件强制访问控制机制融入到该系统中，从而对系统安全提供更高级别的保证。

## 参 考 文 献


[1] Blaze M. A Cryptographic File System for Unix [C] // First ACM Conference on Communication and Computing Security. Fairfax, VA, 1993: 158-165

[2] Cattaneo G, Catuogno L, Del Sorbo A, et al. The Design and Implementation of a Transparent Cryptographic Filesystem for UNIX// Proceedings of the Annual USENIX Technical Conference, FREENIX Track. June 2001: 245-252

[3] Zadok E, Badulescu I, Shender A. Cryptfs: A Stackable Vnede Level Encryption File System. 1998

[4] Howard J H. An Overview of the Andrew File System[C]// Proceedings of the USENIX Winter Technical Conference. Dallas, TX, 1988(2)

[5] Stevens W R. Unix 环境高级编程. 尤晋元，等译. 机械工业出版社，2004

[6] 毛德操，胡系明. Linux 内核源代码情景分析. 浙江大学出版社，2002


(上接第 55 页)

的幅度是 25mV 到 250mV，采样相位是 0～2π。传统采样方案的采样数是 64，新的采样方案的采样数是 128。新的采样方案的平均量化误差是 0.14LSB，传统采样方案的量化误差是 0.38LSB。依据式(2)可以计算出新的采样方案的 ADC 转换器的分辨率达到 8.8bit。

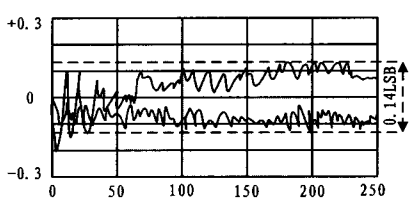


图 6　新的采样方案量化误差范围

**结束语**　本文针对采样时会产生量化噪声干扰，对伺服 burst 信号来说 6 位分辨率的 ADC 转换器显然是不够的，而经过 FFT 计算后 burst 信号会产生较大的误差这一问题，首先依据采样量化误差模型分析了现有的伺服系统过采样模型，并提出了改进型的伺服系统过采样模型。通过仿真得知，改进型采样方案将平均量化误差由原来的 0.38LSB 降到了 0.14LSB，ADC 转换器的分辨率由原来的 7.5bit 提高到 8.8bit，说明该种改进方案是行之有效的。

## 参 考 文 献


[1] Bliss W G, Du L, Karasanbhai M. Digital servo demodulation in a digital read channel. IEEE Transactions on Magnetics, 1998, 34(1): 90-100

[2] Abramovitch D Y. Customizable coherent servo demodulation for disk drives. IEEE/ASME Transactions on Mechatronics, 1998, 3(3): 184-193

[3] 王念旭，等. DSP基础与应用系统设计. 北京：北京航空航天大学出版社

[4] Thapar H, Lee Sang-Soo, Conroy C, et al. Hard Disk Drive Read Channels: Technology and Trends // IEEE Solid States Circuits Society. Proceedings of the IEEE Custom Integrated Circuits Conference. San Diego: Institute of Electronics Engineers, Inc., 1999: 309-316

[5] Grochowski R, Hoyt R F. Future trends in hard disk drives. IEEE Transactions on Magnetics, 1996, 32(3): 1850-1854