

基于组密钥服务器的加密文件系统的设计和实现

肖 达 舒继武 薛 巍 刘志才 郑伟民

(清华大学计算机科学与技术系 北京 100084)

(清华大学信息科学与技术重点实验室 北京 100084)

摘 要 网络存储技术在方便数据共享的同时带来了新的安全隐患,加密文件系统通过密码学方法保证存储在不受用户直接控制的服务器上的文件数据的机密性和完整性. 现有的针对共享加密文件系统的密钥管理方法不能同时满足安全性、灵活性和高效性的需求. 该文提出了加密文件系统 GKS-CFS. 引入可信的组密钥服务器(GKS)集中管理文件加密密钥,GKS 上可以实施灵活的访问控制策略. 通过使用访问控制块和锁盒子,降低了对 GKS 的计算和存储需求,使之可以用硬件实现来增强安全性;通过文件数据的分块加密和密钥版本技术,降低了权限撤销的开销. 作者在 Lustre 上实现了 GKS-CFS 的原型系统并进行了测试. 测试结果表明,由于避免使用了公钥密码算法,和其他系统相比,GKS-CFS 的普通文件操作中的密码学操作开销减少了一个数量级,顺序读写和随机文件操作的性能分别平均降低了 42.0%和 8.4%.

关键词 加密文件系统;机密性;完整性;密钥管理;防损硬件

中图法分类号 TP309

Design and Implementation of a Group Key Server-Based Cryptographic File System

XIAO Da SHU Ji-Wu XUE Wei LIU Zhi-Cai ZHENG Wei-Min

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

(Key Laboratory for Information Science and Technology, Tsinghua University, Beijing 100084)

Abstract Network storage techniques facilitate data sharing but also introduce new vulnerabilities. Cryptographic file systems provide the confidentiality and integrity of file data stored on servers that are not under users' direct control by cryptographic methods. The key management schemes for current shared cryptographic file systems cannot satisfy the security, flexibility and efficiency requirements simultaneously. This paper proposes a cryptographic file system called CKS-CFS. A trusted Group Key Server (GKS) is introduced to manage file encryption keys in a centralized manner and to enable the employment of flexible access control policies. The computation and storage requirement for GKS is reduced through the use of access control blocks and lockboxes so that the function of GKS can be implemented by hardware to provide strong security. The overhead of revocation is reduced by block granularity encryption and key versioning technique. The authors have implemented a prototype of GKS-CFS based on Luster and evaluated its performance. Compared with other systems, the cryptographic cost in common file operations in GKS-CFS is reduced by an order of magnitude by avoiding the usage of public-key cryp-

收稿日期:2007-11-26. 本课题得到国家自然科学基金(60473101)、国家“九七三”重点基础研究发展规划项目基金(2004CB318205)和新世纪优秀人才支持计划项目基金(NCET-05-0067)资助. 肖 达,男,1981年生,博士研究生,研究兴趣包括大规模存储系统和文件系统、存储安全、分布式系统. E-mail: xiaoda99@mails.thu.edu.cn. 舒继武,男,1968年生,教授,博士生导师,研究兴趣包括存储区域网络、并行和分布式计算、算法分析和设计、并行处理技术、集群系统和通信. 薛 巍,男,1974年生,博士,研究兴趣包括电力系统分析和模拟、并行算法设计、集群计算和网络存储. 刘志才,男,1983年生,硕士,研究兴趣包括分布式文件系统和存储安全. 郑伟民,男,1946年生,教授,博士生导师,研究兴趣包括并行计算机体系结构、并行和分布式计算、编译技术、网格计算和网络存储等.

tography; Bonnie++ benchmark test shows that the performance of sequential read/write and random file operations are reduced on average by 42.0% and 8.4% respectively.

Keywords cryptographic file system; confidentiality; integrity; key management; tamper-resistant hardware

1 引言

网络存储技术在方便数据共享和降低管理成本的同时带来了新的安全隐患. 在网络存储环境中, 数据可能存储在不受用户直接控制的服务器上, 传统的访问控制方法已不足以保证用户数据的安全性. 加密文件系统(cryptographic file system)^[1-7]通过客户端的密码学方法保证文件在网络中传输和在服务器上存储过程中的机密性和完整性, 提供端到端的安全性.

加密文件系统可以分为两类: 非共享的和共享的. 非共享加密文件系统(如最早的加密文件系统CFS^[1])不允许文件被多用户共享, 因此不需要考虑加密密钥的管理问题. 但这种系统显然不能满足企业环境中多用户协同工作和数据共享的需要. 在共享加密文件系统中, 允许文件被多个用户共享, 因此一个重要问题是如何管理加密密钥, 使授权用户能够容易得到密钥从而访问文件, 同时未授权用户很难得到密钥.

现有的共享加密文件系统中的密钥管理方法主要分为两类: 一类是将具有相同访问权限的文件分组, 组内文件共享一个密钥, 并由所有者或可信第三方分发给授权用户^[2,6]. 这种方法减少了密钥数目, 但缺点是当一个用户从组中退出从而不再拥有组内文件的访问权限时, 需要将组内所有文件用新密钥重新加密, 计算开销很大. 另一类方法是将文件密钥用每个具有访问权限的用户的公钥加密^[3-4], 这种方法可以实现单个文件粒度的共享, 但只能实现以用户列表方式表示的访问控制策略, 不够灵活.

我们认为, 一个好的密钥管理方案应同时满足三方面需求: (1) 安全性. 密钥的存储和分发应由可信赖的机制来保证; (2) 灵活性. 能够易于实施多种访问控制策略; (3) 高效性. 由加密带来的文件访问和用户权限撤销中的额外开销尽可能小. 现有系统都不能同时满足这些需求.

本文设计并实现了一个新的加密文件系统——GKS-CFS(Group Key Server-Cryptographic File

System), 来实现安全、灵活和高效的文件共享. 该系统针对政府和企业等具有集中控制机构的环境, 引入一个可信的组密钥服务器(Group Key Server, GKS), 集中负责文件密钥的生成和分发. 通过使用锁盒子和访问控制块减少 GKS 的计算和存储需求, 从而有可能以硬件方法确保 GKS 的安全性, 进而保证整个系统的安全性, 同时便于在 GKS 上实施灵活的访问控制. 我们提出一种高效的用户权限撤销机制, 采用了以文件块为粒度的密钥管理策略, 并结合密钥版本技术, 有效降低了权限撤销的开销. 我们在 Lustre^①上实现了 GKS-CFS 的系统原型并进行了测试. 由于避免使用了公钥密码算法, 和其他系统相比, GKS-CFS 的普通文件操作中的密码学开销减少了一个数量级. Bonnie++ 的测试结果显示, 顺序读写和随机文件操作的性能分别降低了 42.0% 和 8.4%.

2 安全模型

本节介绍 GKS-CFS 所采用的假设和安全模型. 系统中有五种角色: 所有者(创建文件和设置访问权限)、读者(对文件有读权限)、写者(对文件有读写权限)、文件服务器(存储文件数据)和组密钥服务器(存储密钥). 所有者、读者和写者通过文件系统客户端操作文件.

在 GKS-CFS 中, 客户端是不可信的, 恶意用户和攻击者可能试图通过客户端访问他们无权访问的文件, 用户存放在客户端上的私密信息(如私钥)也有可能被攻击者窃取. 文件服务器也是不可信的. 它可以将数据泄露给非法用户、篡改数据或向用户提供不真实数据. 系统中唯一可信的是组密钥服务器, 它负责管理用于加密文件的密钥并向合法用户提供密钥. 考虑到在一个实际的企业应用环境中, 海量数据的存储需要有多多个文件服务器, 这些服务器可能隶属于不同的管理域, 保证所有这些服务器的安全是困难的, 相比之下, 保证一个组密钥服务器的安全

① Lustre. <http://www.lustre.org>

是相对容易的,所以我们认为我们的安全假设是合理的.需要指出的是,在 GKS-CFS 中,所有用户需要对组密钥服务器赋予绝对信任,因此不适于无集中控制机构松散耦合的组织,如等网络环境,而适合于具有集中控制机构的组织,如政府、企业等.

GKS-CFS 通过密码学技术保证文件数据的机密性和完整性.文件的内容(可理解的)只会被授权用户获得.当未授权用户或文件服务器试图篡改文件数据时,授权用户能够检测到该行为. CKS-CFS 不提供数据的可用性保证,不能防止恶意用户或服务器删除数据.如需提供可用性,需要将密码学技术和冗余技术结合,这不在本文的讨论范围之内.

3 系统设计

3.1 基本思想和设计目标

在 GKS-CFS 中,GKS 作为全局唯一的可信节点,统一负责维护和分发文件密钥.用户访问每个文件前,都要向 GKS 发出请求,获取该文件的加密密钥和签名密钥(写文件). GKS 首先验证用户身份,并根据预先设置的访问控制策略判断用户是否有对文件的访问权限,如果有,则将文件密钥返回给用户;如没有,则拒绝该请求.

由于所有密钥由 GKS 统一管理,只要保证 GKS 的安全性,就能保证整个系统的安全性;并且 GKS 上可以根据需求实施灵活的访问控制策略.

GKS-CFS 的设计目标是:

(1) 对 GKS 的计算和存储需求最小. GKS 上只需存储少量自身的密钥,而不需要存储每个文件的加密密钥和签名密钥,同时用户访问一个文件的过程中,GKS 只需做少量针对密钥的加解密运算,而不需要对大量文件数据加解密,使 GKS 易于用安全硬件实现,从而保证安全性.

(2) 额外开销尽可能小. 额外开销包括密码学操作的开销和权限撤销时重新加密数据的开销.

3.2 锁盒子和访问控制块

GKS-CFS 中采用的对象及它们之间的关系如图 1 所示. 本文中所用缩略语及其含义在表 1 中列出. 每个文件被分成若干个数据块(block),每个数据块用一个随机生成的对称密钥(数据块密钥, FBK)加密. GKS-CFS 采用了文献[2]引入的锁盒子(lockbox)的思想,一个文件的所有 FBK 存放在该文件对应的锁盒子中,并用该文件的锁盒子密钥(LBK)加密,LBK 也是对称密钥,只有对文件有读

权限的用户才能获取 LBK. 每个 FBK 还对应一个版本号,用于权限撤销时的密钥更新(见 3.5 节). 在锁盒子中和 FBK 一同存放的还有每个数据块的安全 hash 值(cryptographic hash),用以校验数据块的完整性. 将一个文件所有数据块的 hash 值组织成一个 Merkle hash 树^[8],hash 树的根(root hash)用文件签名密钥(FSK)加密后保存在访问控制块中. 只有对文件有写权限的用户才能获取 FSK. 注意本文的 FSK 和文献[3,6]中的文件签名密钥不同,前者是一个对称密钥,后者是一个公私钥对中的私钥.

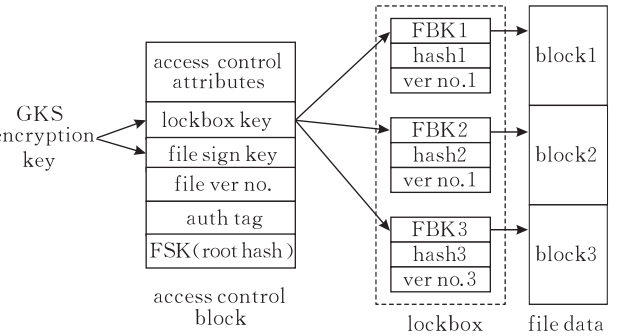


图 1 GKS-CFS 的对象及对象间关系(实线箭头代表对称加密)

表 1 缩略语及其含义

缩略语	含义
FBK	File Block Key
LBK	Lockbox Key
FSK	File Sign Key
GEK	GKS Encryption Key
GSK	GKS Sign Key
ACB	Access Control Block

每个文件对应一个访问控制块(ACB),用以控制授权用户对 LBK 和 FSK 的获取. ACB 中包含文件的访问控制属性(access control attributes)、LBK、FSK、文件版本号、完整性标识(auth tag)和加密的根 hash,其中 LBK 和 FSK 用 GKS 维护的一个加密密钥(GEK)加密,只有 GKS 才能解密. 访问控制属性是由文件所有者创建并由 GKS 检查的文件访问规则,例如一个访问控制列表(ACL),规定了哪些用户和组可以访问文件以及他们的权限. 文件版本号用于权限撤销时的密钥更新(见 3.5 节). 完整性标识是其他域的一个带密钥消息认证码(keyed HMAC),用以防止对访问控制块的篡改. 生成认证码的 GKS 认证密钥(GSK)由 GKS 维护,因此只有 GKS 能根据文件所有者的需要对 ACB 的内容进行修改. 注意加密的根 hash 不参与完整性标识的计算,因为该值在写文件时会发生变化,可以避免每次写文件都由 GKS 重新生成访问控制块.

用户访问文件时,在密钥请求中将 ACB 和自己的身份标识发送给 GKS,GKS 首先用 GSK 验证 ACB 的完整性,然后根据其中的访问控制属性和用户的身份标识确定用户对文件的访问权限,如果用户有读(写)权限,就将 LBK(和 FSK)用 GEK 解密后返回给用户。

GKS-CFS 采用锁盒子和访问控制块的好处是:(1)GKS 不需要维护每个文件的加密密钥,只需要维护两个密钥:GEK 和 GSK,而 FBK 作为元数据的一部分和文件数据一起存放在文件服务器上,减少了 GKS 的存储需求。(2)访问文件时,GKS 只需要做数据量很小的对称密钥加解密(对 LBK、FSK 和根 hash),不需要公钥加解密,减少了 GKS 的计算需求。(3)用户不需要在客户端主机上维护自己的私密信息(如用户私钥),减少了客户端主机被入侵带来的危害。

需要指出的是,一个实际系统可能被划分成多个管理域,每个管理域应有自己的 GKS. 因此在 ACB 中还应包含域 ID,用户访问文件时根据域 ID 找到相应的 GKS. 为简明起见,这部分在上文描述中已省略. 另外,考虑到一旦 GKS 维护的 GEK 丢失,则域内的所有文件都将不可访问,因此必须防止对 GKS 的可用性攻击,这可以通过将 GEK 以秘密分存的方法存储到多个恢复代理上来实现。

3.3 文件访问协议

需用户访问文件时客户端和 GKS 的通信协议如图 2 所示。

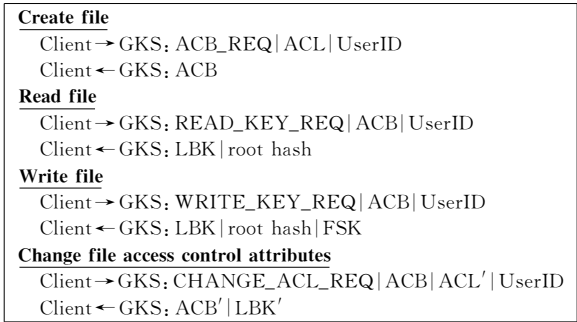


图 2 文件访问协议

创建文件:文件所有者向 GKS 发送获取 ACB 请求,其中包含用户标识和文件的访问控制属性(ACL). GKS 根据 ACL 生成 ACB 返回给所有者. 所有者将 ACB 作为文件元数据的一部分存储到文件服务器上。

读文件:读者首先从文件服务器获取 ACB,然后向 GKS 发送获取读密钥请求,其中包含用户标识

和 ACB. GKS 根据 ACB 中的访问控制属性检查用户权限,然后将 LBK 和根 hash 解密后返回给读者. 读者用 LBK 解密锁盒子,并用锁盒子中的数据块 hash 重新计算根 hash 与收到的根 hash 比较,然后用锁盒子中的 FBK 解密从文件服务器读取的文件数据,并用数据块 hash 校验完整性。

写文件:过程和读文件类似,区别在于 GKS 将文件签名密钥(FSK)和根 hash 一起返回给写者,写者关闭文件时,重新计算根 hash,并用 FSK 加密,替换 ACB 中原有的域。

更改文件访问权限:所有者首先从文件服务器获取 ACB,然后向 GKS 发送更改访问控制属性请求,其中包含用户标识、ACB 和新的访问控制属性. GKS 首先检查用户是否是文件所有者,然后根据新的访问控制属性生成新的 ACB 返回给所有者. 所有者用新的 ACB 覆盖文件服务器上的旧 ACB,并用新的 LBK 重新加密锁盒子。

3.4 身份认证

为了保证 GKS 只响应授权用户的解密请求,GKS 必须能够对发出请求的用户身份进行认证. 用户认证可采用多种方法,如基于对称密钥的 Kerberos 协议^[9]或公钥基础设施(PKI)等. 需要指出的是,GKS-CFS 的主要设计目标是加密文件系统的高效的密钥管理,它本身并不提供认证机制,而需要和一个现有的独立的认证机制协同工作保证安全性. 例如,如采用 Kerberos 认证方式,则需要一个票据分发服务器向用户分发代表身份的票据,同时 GKS 和票据分发服务器共享一个密钥用于验证用户提供的身份票据的真实性. 这种密钥管理机制和认证机制分离的策略提高了系统部署时的灵活性. 本文主要关注基于 ACB 的访问控制和密钥管理方法,具体的认证协议将不作进一步讨论。

在上节描述的文件访问协议中,用户向 GKS 发出请求之前,先要向 GKS 认证身份并协商会话密钥,之后用户和 GKS 间的通信内容都用会话密钥加密保证并用随机数等方法保证完整性. 为清晰起见,图 2 中只有用户和 GKS 间发送的消息内容,省略了消息加密和完整性保证部分。

3.5 权限撤销

用户权限撤销是指使一个原本对文件有访问权限的用户不再具有对该文件的权限. 当从文件的访问控制列表中移除一个用户项,或将一个用户从一个组中剔除时,都会发生权限撤销. 在一般的加密文件系统中,权限撤销操作有较大的计算开销,因为需

要用新密钥重新加密文件,避免被撤销的用户用原有的文件密钥访问文件.在 GKS-CFS 中,为了降低权限撤销的开销,采用了懒惰权限撤销(lazy revocation)的方法,其基本思想是,当一个文件需要重新加密时,推迟重新加密的时机,直到文件内容被更新时,同时只对被更新的部分重新加密,从而通过减少加密的数据量降低计算开销.

在 GKS-CFS 中,通过 LBK 版本号和数据块密钥版本号控制权限撤销中的重新加密过程.创建文件和向新文件写入数据时,LBK 版本号和 FBK 版本号初始化为零.发生权限撤销时,更新文件的 LBK,用新的 LBK 重新加密锁盒子中的 FBK(注意不需要重新加密文件数据块),同时将 LBK 版本号增加 1.此后,对该文件有写权限的用户对文件的某个数据块更新或在文件尾追加新的数据块时,首先比较数据块的 FBK 版本号和 LBK 版本号,如果前者小于后者,表明数据块需要重新加密,则更新 FBK,用新的 FBK 加密新写入的数据块,最后将 FBK 版本号增加到与 FBK 版本号一致,使下一次写入不必再更新 FBK;如果前者等于后者,表明数据块不需要重新加密,用原有 FBK 加密新写入数据块即可.通过文件分块加密和 FBK 版本号的使用,避免了只更新文件的一小部分就需要将整个文件重新加密的情况,最大限度地降低了权限撤销对正常文件访问的影响.

4 安全性分析

本节对 GKS-CFS 系统的安全性进行分析,指出我们的系统可以防止哪些攻击,哪些攻击仍然是可能的,并讨论哪些方法可以防止这些攻击.以下分析均假设文件服务器上的访问控制机制是不可信的,一个非授权用户或攻击者可以通过文件服务器获得加密的文件数据、锁盒子和访问控制块.可以防止的攻击包括:

(1)对文件没有读权限的用户或攻击者读取文件内容.由于文件数据块用 FBK 加密,一个文件的所有 FBK 由 LBK 加密保存在锁盒子中,LBK 由 GEK 加密放在 ACB 中,因此,没有读权限的用户向 GKS 发出请求时,GKS 不会将 LBK 解密返回给用户,从而用户不能用 LBK 解密锁盒子中的 FBK,进而用 FBK 解密文件数据块读取其内容.

(2)对文件没有写权限的用户或攻击者篡改文件数据.没有写权限的用户向 GKS 发出请求

时,GKS 不会将 FSK 解密返回给用户,从而用户不能按其意图更改根 hash 的值.如果用户更改数据块,当另一个授权用户读取文件,重新计算并校验根 hash 时,或者发现数据块和锁盒子中的 hash 值不匹配,或者发现锁盒子中的 hash 值和 ACB 中的根 hash 不匹配,两种情况都表明文件已被非法篡改.

(3)除所有者外的其他用户通过修改 ACB 中的访问控制属性获得相应权限.ACB 具有完整性标识(auth_tag)保证其不可篡改性,只有 GKS 才能根据文件所有者的请求对 ACB 中的任何内容(包括访问控制属性)进行修改.其他用户或攻击者试图更改 ACB 并用更改过的 ACB 向 GKS 发送请求时,GKS 会通过校验 auth_tag 发现 ACB 的完整性已被破坏并拒绝请求.

(4)攻击者通过入侵 GKS 获得 GEK 和 GSK.鉴于整个系统的安全性取决于 GEK 和 GSK 两个密钥的安全性,在真正系统的实现中,应将它们保存在 GKS 上的特殊防损硬件(Tamper-Resistant Hardware, TRH)中,该硬件可由硬件安全模组(Hardware Security Module, HSM)或智能卡(smartcard)实现,使用它们加密和解密 LBK 和 FSK 的操作也都在该特殊硬件内进行,保证这两个密钥在任何情况不会离开该特殊硬件.这样即使 GKS 的软件被攻破,攻击者获得对 GKS 操作系统的完全控制权,他也无法获得 GEK 和 GSK,从而保证保存在文件服务器上的文件仍然是安全的(在本文的实现中,由于条件限制,我们仍然用软件模块实现 GKS).

仍然可能的攻击包括:

(5)对文件服务器的攻击.攻击者可以攻击文件服务器,当攻击者控制文件服务器时,可以恶意损坏或删除其上的文件,造成文件对授权用户不可访问.事实上,用单纯的密码学方法不能防止这种对数据可用性的攻击.可以将本文提出的方法和数据冗余技术相结合提供可用性保证,例如将文件以多副本的形式保存在多台文件服务器上.

(6)对组密钥服务器的攻击.攻击者可以攻击组密钥服务器.虽然不能获得 GKS 密钥(GEK 和 GSK),但可以将 GKS 停机或将保存密钥的硬件拆除,造成保存在文件服务器上的文件不能解密,但文件内容不会泄露.为防止这种攻击,可以将 GKS 密钥以秘密分存的方式备份到多个恢复代理上,以便在存储密钥的硬件丢失时恢复密钥.另一种方法是

设置多台 GKS 同时提供密钥服务。

(7) 对客户端的攻击. 攻击者可以通过攻击文件系统客户端得到当前用户正在访问的文件及其密钥. 但和基于文件组的密钥管理策略^[6]相比, 由于采用了细粒度的以文件为单位的密钥管理策略, 一次成功的攻击只能获取单个文件的密钥, 而不是泄露组内所有文件的共同密钥, 将密钥泄露带来的损害降到了最低。

5 扩展方案

我们在原有系统设计的基础上提出一个扩展方案的设计, 进一步提高系统的可用性和性能. 在原系统设计中, GKS 是系统中唯一的安全访问入口点, 容易形成安全性的薄弱环节和性能的瓶颈. 而扩展方案的基本思想是: 将 GKS 的功能集成到一个卡片上, 称为 GK-Card, 该卡片可以用硬件安全模组 (Hardware Security Module, HSM) 或智能卡 (smartcard) 实现. 为每一个需要使用密钥服务的文件系统客户端安装一个这样的 GK-Card. 当用户需要解密 ACB 中的密钥时, 不再向网络上的 GKS 发出请求, 直接向本机的 GK-Card 发请求即可。

扩展方案和原 GKS-CFS 采用相同的密钥层级结构, 每个 GK-Card 负责加密和解密 ACB 中的相关密钥, 同时负责用户身份认证和访问控制. 系统管理员为每个希望使用加密文件系统的客户端分发一个 GK-Card, 并将 GEK 和 GSK 初始化到 GK-Card 中. 客户端可以利用 GK-Card 完成 ACB 中密钥的加解密, 但不能获得其中的 GEK 和 GSK, 保证了安全性。

和原方案相比, 扩展方案有两个优点: 首先, 如果本机的 GK-Card 出现故障, 只会造成本机上的客户端无法访问文件, 其他机器上的客户端仍然可以通过自己的 GK-Card 正常访问文件, 一个 GK-Card 的丢失也不会造成 GKS 密钥的丢失, 提高了系统的可用性. 其次, 各密钥操作均在客户端完成, 不需要发往 GKS 执行, 避免了和 GKS 建立安全连接的网络通信延迟及链路上用于认证、消息机密性和完整性计算的开销, 提高了系统的性能。

由于每个 GK-Card 只存储 GEK 和 GSK 两个密钥且不需要频繁更新, 因此各个 GK-Card 可以独立行驶功能而不需要互相交互, 保证了方案的可行性. 关于该扩展方案的一些具体问题, 如 GK-Card 的接口和内部设计、文件系统客户端和 GK-Card 间

的通信协议, 还需要做进一步研究, 这也是我们的下一步工作之一。

6 系统实现

为了验证 GKS-CFS 的设计方案的可行性和效率, 我们在 Lustre 上实现了 GKS-CFS 的原型系统. Lustre 是 CFS 公司开发并得到广泛应用的开源并行文件系统, 由文件系统客户端 (FS Client)、元数据服务器 (MDS) 和对象存储设备 (OST) 三部分组成, 每部分又包含若干内核模块. 例如, 在客户端中包含实现文件系统逻辑的模块 llite、和 OST 交互的模块 OSC、和 MDS 交互的模块 MDC、在 OSC 之上构造逻辑对象设备的模块 LOV 等。

GKS-CFS 的软件结构如图 3 所示. 其中白色矩形框代表 Lustre 原有软件模块, 灰色矩形框代表我们新添加的模块. 我们增加了独立的内核模块 GKS 作为组密钥服务器, 同时在客户端添加了两个内核模块 GKC 和 Crypto. GKC 负责和 GKS 通信并向上提供密钥管理接口, 是 3.3 节描述的文件访问协议的执行者. Crypto 模块实际完成文件数据块的加解密操作, 它调用了 Linux 内核提供的 Crypto API 函数库. 创建和打开文件时, 文件系统客户端调用 GKC 提供的接口, 设置内存中和密钥相关的数据结构; 读写文件时, OSC 模块在将一个数据块写入 OST 之前或从 OST 读出一个数据块之后调用 Crypto 提供的接口加密或解密数据块. 为简明起见, 图中只显示了 Lustre 的客户端中和主要 I/O 逻辑联系紧密的模块, 而省略了其他模块。

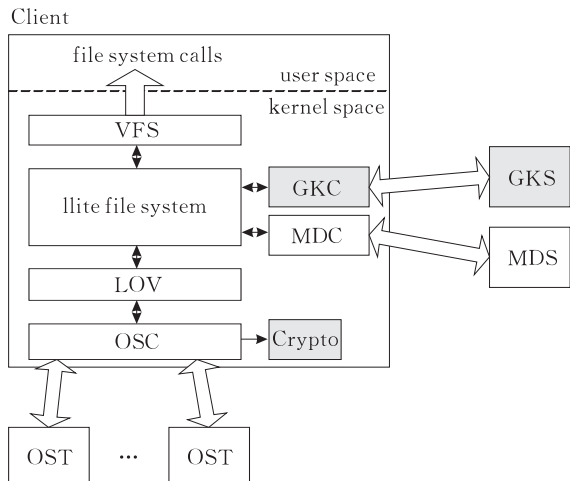


图 3 GKS-CFS 的软件结构

文件的锁盒子以扩展属性的形式保存在 MDS

上. 在当前实现中, GKS 没有对 GKC 发出的请求进行用户身份认证, 并且消息是以明文传输的. 以后计划引入 Kerberos 认证机制, GKS 通过 GKC 提供的票据认证其身份, 并通过 GKS 和 GKC 协商的会话密钥保证二者间消息的机密性和完整性.

7 性能评价

我们在 GKS-CFS 的原型系统上进行了测试, 以衡量加密操作对系统性能的影响. 测试内容包括加解密相关操作的开销和给予基准测试程序的读写性能测试.

7.1 密码学操作开销

表 2 列出了各种文件操作中加解密相关操作的额外开销. 测试服务器的配置为 Intel Pentium III x4 处理器, 1GB 内存, 运行 Redhat Linux 操作系统(内核版本为 2.6.10). 各种加解密相关操作的开销用 Linux 内核提供的 Crypto API 库函数测量. 在本文所有测试中, 数据块大小为 4KB, 数据块用 128 比特密钥长度的 AES 算法加密, 加密模式为计数器模式. Hash 值的计算采用 SHA-1. 为了获得计算根 hash 和解密锁盒子的开销, 假设文件的平均大小为 2MB. LBK、FSK 和根 hash 同样使用 128 比特密钥长度的 AES 加密, 模式为 ECB. ACB 的默认长度是 512 字节.

表 2 加解密相关操作的开销

文件操作	加解密相关操作	延迟/ms	合计/ms	执行者	频率
创建文件	加密 LBK 和 FSK	0.19	0.51	GKS	每个文件
	计算 ACB 的认证码	0.32			
打开文件	校验 ACB 的认证码	0.32	0.70	GKS	每个文件
	解密 LBK 和 FSK	0.19			
	解密根 hash	0.19			
	校验根 hash	1.52	3.47	读/写者	
	解密锁盒子	1.95			
关闭文件	计算根 hash	1.52	1.71	读/写者	每个文件
	加密根 hash	0.19			
写文件	加密数据块	1.04	1.93	写者	每个数据块
	计算数据块 hash	0.87			
读文件	解密数据块	1.05	1.92	读者	每个数据块
	校验数据块 hash	0.87			

从表 2 可以看出, 打开文件的主要开销是计算根 hash 和解密锁盒子, 关闭文件的主要开销是校验 hash 根. 读/写文件的开销是加/解密数据块和计算/校验数据块 hash. 权限撤销操作和创建文件类似, 只是增加了重新加密锁盒子的操作, 为简明起见, 它的开销没有在表 2 中列出. 当读写大文件时, 加解密数据块和计算/校验签名将占额外开销的主要部分.

下面我们将 GKS-CFS 和基于公私钥的密钥管理方案, 在创建和打开文件时的文件密钥操作的相关开销进行比较. 在基于用户公私钥的密钥管理方案(如文献[3, 5-6])中, 创建文件时需要将文件密钥用每个具有权限的用户的公钥加密, 并对密钥元数据签名, 相关开销为 $T_{create} = n \cdot T_{PubEnc} + T_s$, 打开文件时需要用用户私钥解密文件密钥并验证密钥元数据的签名, 相关开销为 $T_{open} = T_{PubDec} + T_v$. 其中 T_{PubEnc} 和 T_{PubDec} 分别为一次公钥加密和公钥解密的时间, T_s 和 T_v 分别为一次签名操作和验证签名操作的时间, n 为对文件具有权限的用户数目. 在 GKS-CFS 的基于对称密钥的密钥管理方案中, 创

建文件的相关开销为 $T_{create} = 2 \cdot T_{SymEnc} + T_{MAC}$, 打开文件的相关开销为 $T_{open} = 2 \cdot T_{SymDec} + T_{MAC}$, 其中 T_{SymEnc} 和 T_{SymDec} 分别为一次对称密钥加密和解密的时间, T_{MAC} 为一次计算对 ACB 计算认证码的时间. 公私钥密码的计算复杂度要大大高于对称密码的计算复杂度. 在我们的测试平台上测得一次公钥加密和解密的延迟分别为 50ms 和 15ms 左右(签名和验证操作的延迟在同一个数量级), 大大高于 AES 加解密和计算认证码的延迟. 相比其他系统, 由于我们提出的方法中没有使用公钥密码系统, 使普通文件操作中的密码学操作开销减少了一个数量级. 另外我们看到, 由 GKS 执行的操作只在创建和打开时发生且开销都很小, 大部分操作由客户端完成, 这满足了设计目标中尽量降低 GKS 计算量的需求.

7.2 IOZone 性能测试

我们用 IOZone 测试工具对 GKS-CFS 原型系统的性能进行了测试, 并不带加密功能的基准 Lustre 系统进行了比较, 测试环境由百兆以太网连接的 4 台服务器组成, 分别充当客户端、MDS、OSD

和 GKS,服务器配置同 7.1 节. 我们首先测试了加解密计算带来的额外开销. 为了更准确地测量出加解密计算的开销,我们将文件大小设为系统内存的 2 倍即 2GB,来尽量屏蔽文件系统缓存的影响,测试了在不同 I/O 请求大小下的吞吐率变化. 结果如图 4 和图 5 所示.

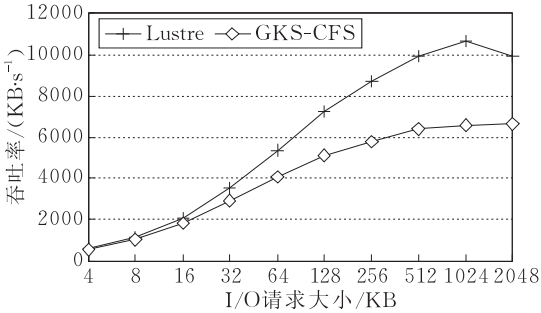


图 4 随机读性能比较

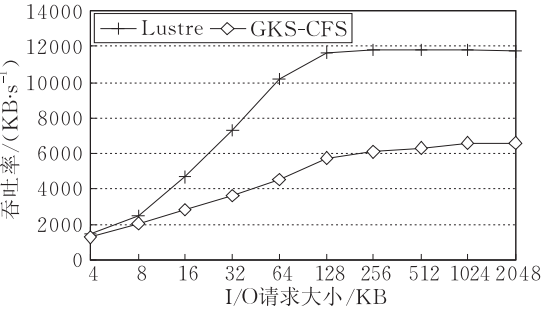


图 5 随机写性能比较

从图 4 和图 5 可以看出,加密部分对吞吐率的影响随 I/O 请求大小的增加而增大,这是因为加解密的延迟随 I/O 请求大小的增长速度基本是线性的,而由于受寻道时间的影响,I/O 的响应时间随 I/O 请求大小的增长速度低于线性,因此加密延迟在总延迟的比例逐渐增大. 当 I/O 请求大小为 64KB 时,随机读和随机写的吞吐率分别降低了 23% 和 56%.

接下来我们测试了 GKS-CFS 在不同文件大小下的随机读写和顺序读写吞吐率并和基准 Lustre 系统比较. 文件大小的变化范围是 64KB~256MB, I/O 请求大小设为 64KB. 顺序读写和随机读写的结果分别如图 6 和图 7 所示. 从两图中可以看出,当文件大小小于 4MB 时,GKS-CFS 和 Lustre 的吞吐率差别较小,不超过 14.6%. 这是由于文件较小时,会被缓存在客户端文件系统缓存中,不会触发加解密操作. 当文件大小超过 4MB 而小于 256MB 时,随机写和顺序写的性能会有较多降低,这是由于文件系统客户端 llite 会定期将内存中足够多的脏页面写回到 OST,从而触发加解密操作.

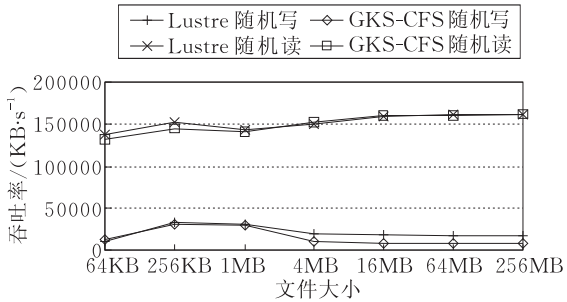


图 6 不同文件大小顺序读写性能比较

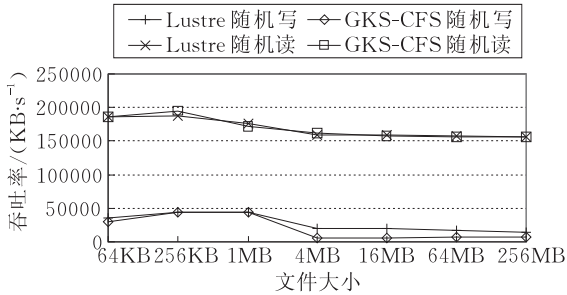


图 7 不同文件大小随机读写性能比较

7.3 Bonnie++性能测试

为了评价 GKS-CFS 在各种文件系统操作(创建、读写、删除、获取属性等)时的性能,我们用 Bonnie++ 基准测试程序进行了测试,文件大小设为内存的 2 倍即 2GB. 测试分两组,第一组是各种模式下的顺序读写性能,共测试 5 种模式:逐字符写(Write Per Chr)、块写(Write Block)、读后写(Rewrite)、逐字符读(Read Per Chr)、块读(Read Block),结果如图 8 所示. 从结果可以看出,顺序读写的性能下降较大,平均降低 42.0%,由于顺序读写时磁盘寻道带来的延迟缩短,因此加解密计算的额外开销占了主导地位. 其中,读后写模式吞吐率下降最大,为 66.8%,这是因为该模式在读出数据和写入数据时需要做两次加解密操作.

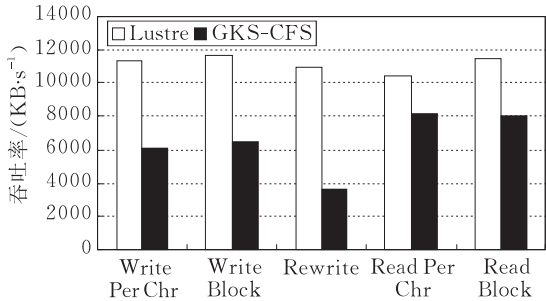


图 8 不同模式顺序读写性能比较

第二组测试是各种随机文件元数据操作的性能,包括创建(Create)、获取属性(Stat)、删除>Delete)、寻址(Seek) 4 种操作,指标是每秒操作次数. 结果如图 9 所示. 从结果可以看出,和顺序读写

相比,随机操作的性能下降较小,平均下降 8.4%,其中创建操作由于需要和 GKS 通信获取密钥,性能下降最大,为 29.1%. 在随机操作中,磁盘寻道的延迟变大,占总延迟的主要部分,而加解密计算开销的影响相对变小.

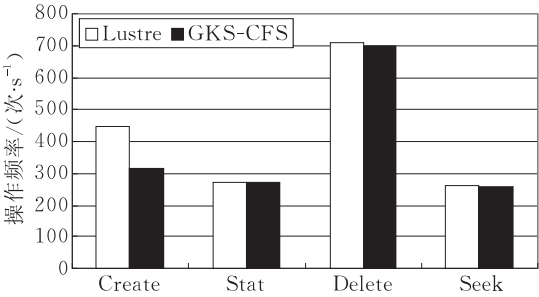


图 9 随机文件操作性能比较

从 Bonnie++ 的两组实验结果可以看出,当应用特点为大数据量连续读写时,GKS-CFS 的加解密对系统性能影响较大;当应用特点为小数据量频繁文件元数据操作时,GKS-CFS 和基准 Lustre 系统的性能差别很小.

7.4 扩展系统的模拟实现和测试

我们基于 GKS-CFS 的工作给出扩展系统的软件模拟实现和性能测试. 由于硬件条件的限制,在实现中我们用软件模拟 GK-Card 的硬件行为,即把 GK-Card 实现为一个内核模块加载到客户端,把 GKS-CFS 系统中 GKS 的密钥操作方法抽取出来封装在模块中,通过接口向客户端文件系统提供密钥操作服务.

我们同样使用 Bonnie++ 测试了系统的文件元数据操作性能,并和基准 Lustre 以及 GKS-CFS 进行比较. 测试环境由 3 台服务器组成,分别充当客户端、MDS 和 OSD,服务器配置同 5.1 节. 我们假设 GK-Card 由硬件安全模组(HSM)实现,通过 PCI 接口和主机连接,接口带宽为 133MB/s. 在我们的实现中,ACB 的长度为 128 字节,则文件系统客户端和 GK-Card 间的传输延迟为 $128\text{B}/(133\text{MB}\cdot\text{s}^{-1})\times 2=1.92\mu\text{s}$. 我们在 GK-Card 端用软件方法模拟这一延迟. 假设 GK-Card 的处理速率是主机 CPU 的五分之一,我们用模块中的每种密钥操作循环 5 次来模拟这种效应. 测试结果如图 10 所示.

从结果可以看出,除文件创建以外,基于 GK-Card 的扩展系统在文件元数据操作性能上与基准 Lustre 性能差别很小. 然而在创建文件时需要创建密钥且对其进行加密,因此比基准 Lustre 对应的性能有所下降. 值得注意的是,扩展系统的随机文件创建操作频率比 GKS-CFS 分别提高了 21.1%和

22.8%,这是因为前者在创建密钥后直接在客户端进行加密,而 GKS-CFS 需要将密钥发送到组密钥服务器加密后再返回给客户端,造成一定的网络通信延迟. 由此可见,扩展系统通过消除网络上的密钥操作流节省了通信开销,降低了密钥操作对系统性能的影响.

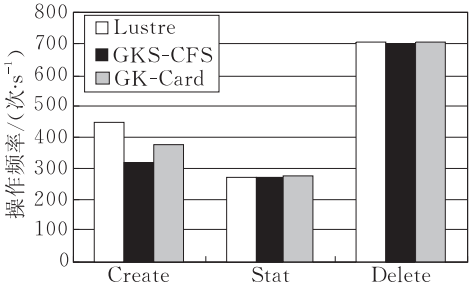


图 10 扩展系统随机文件操作性能测试

8 相关工作

随着网络存储的普及,关于安全存储的研究近年来越来越受到重视,文献中已提出很多加密文件系统,如 Cepheus^[2]、Plutus^[6]、SiRiUS^[3]、NCryptfs^[7]、SSFS^[5]等. 但这些系统的密钥管理机制都不能同时满足安全、高效和灵活性的需求.

Plutus^[6]采用用户维护和分发密钥的方法,并将相同权限的文件划分成文件组(filegroup)并共享一个密钥,以减少密钥数目. 由于访问控制和密钥分发由文件所有者完成,用户可以实施灵活的访问控制策略. 但手动分发密钥给用户带来额外负担,另外权限撤销虽然采用了懒惰的方法,但由于需要重新加密组内的所有文件,开销仍然较大. 攻击者如果通过攻击主机或网络获得文件组密钥,可以访问组内的所有文件. SiRiUS^[3]将文件密钥用每个具有访问权限的用户的公钥加密,这种方法只能实现以用户列表方式表示的访问控制策略,并且创建文件的计算开销(公钥加密操作的次数)随合法用户数目的增加而增大. 攻击者一旦通过攻击主机获得用户私钥,可以访问用户有权访问的所有文件. GKS-CFS 通过引入 GKS 集中管理文件密钥,在 GKS 上可以灵活实施各种访问控制策略,创建文件具有较小的恒定计算开销,权限撤销的开销也通过分块加密降到最低. 攻击者攻破主机只能获得用户当前正在访问的文件,对其他文件没有影响,安全性也得到保证. 文献[10]研究了在网络附加存储(NAS)中通过密码学方法保证文件安全性的问题,提出了一种基于权能(capability)的密钥分发方案来减少存储服务

器的密钥管理负担,但它仍然采用用户公私钥对的机制,并且没有涉及权限撤销时的密钥更新及重新加密问题。

和我们的工作最为接近的是 Hughes 等人提出的 SSFS^[5]. 用绝对可信的组服务器集中管理文件密钥和将访问控制信息和加密后的文件密钥结合组成访问控制块的基本思想是一致的. 但 SSFS 中仍然使用公钥密码系统,为了确保安全性,依赖智能卡存储用户私钥,这限制了系统的应用范围. 另外,SSFS 不提供文件完整性保证,并且加密以单个文件为粒度,没有考虑权限撤销的问题. GKS-CFS 通过将数据块的 hash 值组织成 Merkle hash 数并用文件签名密钥加密树根保证完整性,并提出了使用数据块密钥和密钥版本号的高效的权限撤销机制。

GKS-CFS 使用 GKS 上的可信硬件进行安全的密钥管理. 可信硬件由于其相对于纯软件的方案的优势,在安全领域正在得到越来越广泛的应用. 例如,文献[11]研究了利用 TPM 提供离线存储上的数据的完整性问题,微软的 Bitlocker^① 作为 Windows Vista 操作系统的一个实用工具,通过 TPM 和磁盘加密技术相结合保护用户的敏感文件的内容. 本文中 GKS 上的可信硬件和一般 TPM 的不同是通过访问控制块技术,它将密钥的存储、加解密和访问控制功能集成在一起。

9 结论和下一步工作

本文详细描述了 GKS-CFS 的设计原理和实现方法. GKS-CFS 是一个采用组密钥服务器进行密钥管理的共享加密文件系统,满足了对加密文件系统的密钥管理机制的安全性、灵活性和高效性的需求. 测试结果显示 GKS-CFS 在提高安全性的同时,引入的性能开销可以接受。

下一步计划研究文件名和目录名等元数据的加密机制,因为这些数据也能向攻击者透露有用信息或帮助其确定攻击目标. 下一步工作还包括改进锁盒子的表示方法,以降低和锁盒子相关的计算和存

储额外开销,以及扩展方案的详细设计和实现。

参 考 文 献

- [1] Blaze M. A cryptographic file system for UNIX//Proceedings of the 1st ACM Conference on Communications and Computing Security. Fairfax, Virginia, USA, 1993: 9-16
- [2] Fu K. Group sharing and random access in cryptographic storage file system [Master dissertation]. Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, USA, 1999
- [3] Goh E, Shacham H, Modadugu N, Boneh D. SiRiUS: Securing remote entrusted storage//Proceedings of the 10th Network and Distributed Systems Security Symposium (NDSS'03). San Diego, California, USA, 2003: 131-145
- [4] Halcrow M A. eCryptfs: An enterprise-class cryptographic file system for Linux//Proceedings of the 2005 Linux Symposium. Ottawa, Canada, 2005: 201-218
- [5] Hughes J P, Feist C J. Architecture of the secure file system//Proceedings of the 8th IEEE Symposium on Mass Storage Systems. San Diego, USA, 2001: 277-290
- [6] Kallahalla M, Riedel E, Swaminathan R, Wang Q, Fu K. Plutus: Scalable secure file sharing on entrusted storage//Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST'03). San Francisco, CA, USA, 2003: 29-42
- [7] Wright C P, Martino M C, Zadok E. Ncryptfs: A secure and convenient cryptographic file system//Proceedings of the USENIX Annual Technical Conference. San Antonio, Texas, USA, 2003: 197-210
- [8] Merkle R C. A digital signature based on a conventional encryption function//Proceedings of Advanced in Cryptology-CRYPTO'87. LNCS293. Springer Verlag, 1988: 369-378
- [9] Neumann B C, Ts'o T. Kerberos: An authentication service for computer networks. IEEE Communications, 1994, 32 (9): 33-38
- [10] Zhu Y, Hu Y. SNARE: A strong security scheme for network-attached storage//Proceedings of the 22nd International Symposium on Reliable Distributed Systems (SRDS'03). Florence, Italy, 2003: 250-259
- [11] Dijk M, Rhodes J, Sarmenta L F G, Devadas S. Offline entrusted storage with immediate detection of forking and replay attacks//Proceedings of the 2nd ACM Workshop on Scalable Trusted Computing (STC'07). Alexandria, Virginia, USA, 2007: 41-48



XIAO Da, born in 1981, Ph.D. candidate. His current research interests include mass storage system and file systems, storage security and distributed systems.

SHU Ji-Wu, born in 1968, professor, Ph. D. supervisor. His current research interests include storage area networks, parallel and distributed computing, algorithm analysis and design, parallel processing techniques, and cluster systems and communication.

XUE Wei, born in 1974, Ph. D.. His current research

① BitLocker. <http://www.bitlocker.com/>

interests include power system analysis and simulation, parallel algorithm design, cluster computing and network storage.

LIU Zhi-Cai, born in 1983, M. S.. His research interests include distributed file systems and storage security.

Background

This paper addresses the problem of key management in cryptographic file systems. A secure and efficient key management scheme is the one of the most important factors to put cryptographic file systems into practical use. However, this problem is not addressed well by existing solutions. Two key management schemes have been proposed: The file group based scheme and the user public/private key based scheme. But neither of them can satisfy the security, flexibility and efficiency requirements simultaneously. They either suffer from poor performance due to the use of public-key cryptography or fail to provide flexibility and security due to the coarse-grained key management based on file groups.

This paper proposes a key management scheme for cryptographic file system based on a trusted group key server (GKS) that securely and flexibly manages keys in a centralized manner. We also present the design and implementation of a cryptographic file system call GKS-CFS that adopts this scheme. In the design of GKS-CFS, we make efforts to reduce the computation and storage requirement for GKS so that the function of GKS can be implemented by tamper-resistant hardware, guaranteeing strong security. Efficiency is also achieved by avoiding the usage of public-key cryptography.

The work of this paper is part of the project "Study of the on-demand deployment model and quality of service

ZHENG Wei-Min, born in 1946, professor, Ph. D. supervisor. His research interests include parallel computer architecture, parallel and distributed computing, compiler techniques, grid computing and network storage.

(QoS) for the next-generation internet-based storage" supported by the National Grand Fundamental Research 973 Program of China under grant No. 2004CB318205. This project aims to provide theoretic models and key techniques for the construction of the next-generation networked storage.

Our research group has been conducting research on several areas related to storage QoS, which include data distribution, storage virtualization and storage management, storage security, etc. Our group has proposed techniques and developed systems to enhance various aspects of QoS for storage, including performance, reliability, flexibility, manageability and security. We have published a number of high-quality papers in these research areas and some of the systems developed by us have been used by governments and enterprises as their information infrastructure, such as the Beijing police office and the National Audit Office, etc.

Research on cryptographic file systems is an important area in the field of storage security, which provides confidentiality and integrity for data stored on entrusted storage by using cryptographic techniques. The work of this paper contributes to our research on storage security by presenting key techniques for key management in cryptographic file systems and by designing and implementing a real cryptographic file system based on our techniques.