软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2016,27(9):2400−2413 [doi: 10.13328/j.cnki.jos.004862]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向真实云存储环境的数据持有性证明系统*

肖 达[1,2], 杨绿茵[1,2], 孙 斌[1,2], 郑世慧[1,2]

[1](北京邮电大学 计算机学院,北京 100876)

[2](灾备技术国家工程实验室(北京邮电大学),北京 100876)

通讯作者: 肖达, E-mail: xiaoda99@bupt.edu.cn

摘 要: 对数据动态更新和第三方审计的支持的实现方式是影响现有数据持有性证明(provable data possession, 简称 PDP)方案实用性的重要因素.提出面向真实云存储环境的安全、高效的 PDP 系统 IDPA-MF-PDP.通过基于云存储数据更新模式的多文件持有性证明算法 MF-PDP,显著减少审计多个文件的开销.通过隐式第三方审计架构和显篡改审计日志,最大限度地减少了对用户在线的需求.用户、云服务器和隐式审计者的三方交互协议,将 MF-PDP 和隐式第三方审计架构结合.理论分析和实验结果表明:IDPA-MF-PDP 具有与单文件 PDP 方案等同的安全性,且审计日志提供了可信的审计结果历史记录;IDPA-MF-PDP 将持有性审计的计算和通信开销由与文件数线性相关减少到接近常数.

关键词: 数据持有性检查;第三方审计;显篡改审计日志;存储安全;同态认证元

中图法分类号: TP309

中文引用格式: 肖达,杨绿茵,孙斌,郑世慧.面向真实云存储环境的数据持有性证明系统.软件学报,2016,27(9):2400−2413. http://www.jos.org.cn/1000-9825/4862.htm

英文引用格式: Xiao D, Yang LY, Sun B, Zheng SH. Provable data possession system for realistic cloud storage environments. Ruan Jian Xue Bao/Journal of Software, 2016,27(9):2400−2413 (in Chinese). http://www.jos.org.cn/1000-9825/4862.htm

Provable Data Possession System for Realistic Cloud Storage Environments

XIAO Da[1,2], YANG Lü-Yin[1,2], SUN Bin[1,2], ZHENG Shi-Hui[1,2]

[1](School of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876, China)

[2](National Engineering Laboratory for Disaster Backup and Recovery (Beijing University of Posts and Telecommunications), Beijing 100876, China)

Abstract: The methods for supporting dynamic data updates and third-party audit are key factors that affect the practicality of existing provable data possession (PDP) schemes. This article proposes a secure and efficient PDP system called IDPA-MF-PDP for realistic cloud storage environments. The cost of auditing multiple files is dramatically reduced by a multiple-file PDP scheme based on the data update pattern of cloud storage. The requirement for users being online is reduced to the maximum extent by the implicit third-party audit framework and tamper-evident audit logs. The tripartite interaction protocol between the user, the cloud server and the implicit auditor combines MF-PDP with the implicit third-party audit framework. Theoretical analysis and experimental results show that IDPA-MF-PDP has equivalent security property with single-file PDP schemes and the audit log provides a trustworthy history record of audit results; IDPA-MF-PDP reduces the computation and communication overhead of data possession auditing from linear in the number of files to near constant.

Key words: provable data possession; third-party audit; tamper-evident audit log; storage security; homomorphic authenticator

* 基金项目: 国家自然科学基金(61202082); 国家 242 信息安全计划(2014A120)
Foundation item: National Natural Science Foundation of China (61202082)
收稿时间: 2014-05-07; 修改时间: 2014-12-31; 采用时间: 2015-05-15; jos 在线出版时间: 2015-09-01
CNKI 网络优先出版: 2015-09-01 16:00:47, http://www.cnki.net/kcms/detail/11.2560.TP.20150901.1600.001.html

随着云存储的日益普及,如何保证存放在云端的用户数据的完整性,已成为云计算安全的一个重要问题[1].数据持有性证明(provable data possession,简称 PDP)方案[2]可以使用户远程检查存放在云存储服务器上数据的完整性,且无需将数据下载到本地.虽然近年来已有众多 PDP 方案及其变种被提出[3−10],但这些理论方案距离在真实云存储环境中实际部署尚有距离.有两个关键因素严重影响 PDP 方案的实用性.

- 首先,在云存储环境中,用户不但需要获取远程存储的数据,还应能对这些数据进行更新.虽然研究者已提出支持动态更新操作的 PDP 方案[3−7],但它们都引入了复杂的数据结构,增加了方案的复杂度和审计开销;
- 其次,为了减轻用户负担,现有 PDP 方案多借助独立于云存储服务提供商的机构作为可信第三方(trusted third party,简称 TTP)代替用户执行数据持有性检查[6,8−10].这种架构增加了系统的部署和运营成本,不仅需使用支持公开验证的 PDP 方案[6],还引入了如何防止第三方窃取用户关键数据等新问题.关于如何实现和部署 TTP 以及如何将审计结果以可信的方式提供给用户,现有研究均未给出具体可行的方案.

本文针对动态数据更新和基于 TTP 的 PDP 的实现和部署方式这两个问题给出了切实可行的解决方法,在此基础上,提出面向真实云存储环境的 PDP 系统 IDPA-MF-PDP,为 PDP 从理论到实用迈出了一步.具体来说:

1) 针对动态数据更新问题,我们观察到,云存储有别于传统存储(如文件系统)的数据更新模式,即:典型的数据更新操作不是对文件内容的插入或修改,而是静态文件数量的变化(通常为增加).针对这种模式,我们定义了新的 PDP 方案——多文件数据持有性证明(multiple-file provable data possession,简称 MF-PDP).与原始 PDP 方案不同,MF-PDP 在一次挑战中检查由数目可变的静态文件组成的文件组的完整性,以显著减少审计开销.在现有单文件 PDP 方案的基础上,通过将同态认证元与数据块虚拟下标结合,构造一个安全高效的 MF-PDP 方案;
2) 针对真实云存储环境中 TTP 难以实现或部署的问题,我们提出基于隐式 TTP 的数据持有性审计架构,利用与云服务器集成在一起的防篡改可信硬件作为隐式数据持有性审计者(IDPA)执行 TTP 职能.通过云服务器和隐式第三方的交互,生成可以被任何一方验证的显篡改审计日志,最大限度地降低对用户在线时间的需求,用户可以随时通过查看日志,以离线的方式审计云端数据的完整性.这种实现方式有效地降低了 TTP 的部署和运营成本,且由于 TTP 是部署在云端的可信硬件,避免了第三方审计者窃取用户隐私数据的问题;
3) 为了将 MF-PDP 与隐式 TTP 审计架构集成,我们提出用户、云服务器和隐式第三方间的三方交互协议.在交互中完成 MF-PDP 的挑战-应答过程以及审计结果的可信生成和保存,形成完整的 IDPA-MF-PDP 系统.理论分析和实验结果表明:IDPA-MF-PDP 具有和单文件 PDP 方案等同的安全性,且审计日志提供了可信的的审计历史记录;IDPA-MF-PDP 将持有性审计的计算和通信开销由线性减少到接近常数水平,其性能主要受限于硬盘 I/O 能力,而非密码运算.

1 多文件数据持有性证明

1.1 数据更新模型

我们观察到,云存储中的数据更新模型与传统存储系统不同:在传统文件系统中,文件内容常被读写、修改或删除;但是在云存储中,文件对象一旦被写入云中就会保持相对静态,即,用户很少修改、新增或者删除文件本身的内容.在云存储的两个主要应用——备份和归档中,写入云中的数据,如文件系统的快照或者归档的对象,本身就是静态的.云存储中数据更新的另一个特点是:删除操作比较少发生,并且通常以一组文件为单位发生(例如,一个到期归档对应的一组对象).

基于以上观察,我们提出了一个适用于云存储的数据更新模型:将一个由多个文件组成的文件组作为数据更新的基本单元.向文件组中增加新的文件,即为文件组的更新操作.文件被添加到文件组后,就不能再被修改或者从文件组中删除.删除操作以文件组为单位进行,一个文件组中的所有文件被当作整体一并删除.在该模型

基础上,我们提出 MF-PDP 方案.

下面对该云存储数据更新模型的提出依据和成立范围做进一步说明:

- 首先,从统计数据上,欧盟统计局提供的 2014 年个人使用云存储情况调查报告[11]中统计了使用云存储服务存储各种文件类型的人口百分比.静态文件(照片、音乐、视频)对应的人口百分比之和(144%)显著高于动态文件(文档、电子表格、演示文稿)对应的人口百分比之和(54%).考虑到普通个人用户产生的上述静态文件的大小一般都大于上述动态文件,可以估计如果按照文件总大小统计,云存储中的静态文件的比例会更高.
- 其次,从云存储自身的定义上,狭义的云存储特指底层云存储服务,如 Amazon S3,Google Cloud Storage,Microsoft Azure Blob storage 等;而广义的云存储还包括面向终端用户的在线文件备份同步和笔记类服务,如 Dropbox、Mozy、金山快盘、印象笔记等.虽然后者提供文件修改操作接口,但前者一般只提供不可变对象操作接口,即,只允许对一个完整的对象通过网络响应进行读、写、删除操作.主流云存储服务 Amazon S3 和 Google Cloud Storage 提供的都是这种接口.本文的数据持有性证明方案主要针对这种底层云存储服务,作为基础安全设施.因此,采用该数据更新模型是合理的.

值得指出的是:面向终端用户的在线文件备份服务可构建在云存储服务之上,并可将文件修改操作转化为不可变对象的添加.例如,主流在线文件同步服务提供商 Dropbox 就是用 Amazon S3 作为其后端存储,并用二进制差量比对算法 librsync[12]将文件更新转化为新文件相对旧文件的增量块存储在 S3 上.开源在线备份工具 Tarsnap[13]采用类似的做法.学术界以云存储服务作为后端存储的文件系统,如 Cumulus[14]和 BlueSky[15]等,对外提供符合 POSIX 语义的标准文件系统接口,但在内部将文件切分为定长或可变长的分块作为不可变对象保存到 Amazon S3 中,以实现重复数据删除.在这些应用场景中,数据更新模型中所指的文件组中的“文件”并不是一般意义上的文件,而是文件拆分后形成的不可变对象(数据块).

1.2 MF-PDP方案

MF-PDP 模型包含 5 种多项式时间算法.

- ***KenGen***$()\rightarrow(pk,sk)$是一个密钥生成算法.用户和验证端 DPA 都可用此算法生成公私钥对,自己保存私钥,并把公钥分发给其他两方.我们用 *upk*,*usk* 表示用户公私钥对,用 *dpk*,*dsk* 表示 DPA 公私钥对;
- ***Add***$(usk,F,\alpha,GID)\rightarrow(F',M,\alpha')$是一个向文件组添加文件的算法,由用户运行.算法将文件 F 分块,每块长度为 L 位,为每个文件块生成认证元数据,并更新持久状态α(α为验证端为每个文件组维护的持久状态).输出为文件块集合 F',认证元数据 M,及更改后的持久状态α';
- ***Challenge***$(\alpha)\rightarrow chal$ 是为当前文件组生成挑战的算法,由验证端运行.算法以该文件组持久状态α为输入;
- ***Prove***$(upk,chal,F',M,\alpha)\rightarrow P$ 由服务器端运行,为挑战生成相应的证据.算法以用户公钥 *upk*、挑战 *chal*、一组文件块 F'这些文件块对应的认证元数据 M 及文件组α为输入,输出证据 P;
- ***Verify***$(upk,chal,P,\alpha)\rightarrow\{0,1\}$是由验证端运行验证证据 P 的算法.以用户公钥 *upk*、挑战 *chal*、证据 P 和持久状态α为输入.如果验证通过输出“1”,否则输出“0”.

上述算法和文献[2]中 PDP 模型算法的主要区别为:(1) 增加了向文件组添加文件的 Add 算法;(2) 算法 Challenge,Prove 和 Verify 针对一个文件组而非一个文件.

一个原始的 MF-PDP 方案可以通过对文件组中每个文件使用某种单文件 PDP 方案(如文献[2])来实现,显然,该方案的复杂度为 $O(n)$,其中,n 为文件组中文件的个数.下面给出我们在文献[16]中提出的一个通过集成挑战来实现的更加高效的 MF-PDP 方案,其基本思想是:将这个文件组虚拟成为一个文件,将向文件组中添加文件的操作看作对虚拟文件的追加.这样,对这个虚拟文件的一次挑战就可以检查这个文件组中的所有文件.在这个虚拟文件中,每个文件块都有一个虚拟下标用作标识,这些下标也被用来生成同态认证元(homomorphic authenticators).

需要指出:运行于底层云存储层的 MF-PDP 只是提供了文件分组机制,具体的文件分组策略需要由使用云

存储服务的上层应用根据应用需求确定,使组内文件具有逻辑上的相关性,以方便文件管理.例如:在法规遵从应用中,可将需要保存相同年限的文件分为一组;在归档应用中,可将同一文件系统卷在不同时间点的归档切分后形成的数据块分为一组;在照片备份应用中,可将用户在某一时期的照片文件分为一组.由于在文件总数一定的条件下,持有性检查的计算开销和 DPA 的存储开销都和文件组个数成正比,因此在满足应用需求的前提下,应尽量减少文件组的个数,即,使每个文件组内包含尽量多的文件.

图 1 给出 MF-PDP 方案的算法描述.MF-PDP 的运行过程与基于随机抽样的单文件 PDP[2]相似:在挑战阶段,验证端 DPA 向云存储服务器 SSP 发起挑战,验证通过虚拟下标标识的 c 个文件块的持有性,c 个下标随机分布在文件组的所有文件中;SSP 向 DPA 返回这 c 个块和其认证元的聚合值,DPA 对这两个值进行验证.算法用到 3 个密码学原语:哈希函数 H、伪随机函数(PRF)e 和伪随机置乱函数簇(PRP)$\pi[T]$,分别用于生成同态认证元、聚合数据块和认证元的线性系数和抽查数据块的虚拟下标.文件组持久状态α中包含当前文件组的大小(用 B 表示),在文件组创建过程中被初始化为 0.关于该方案算法的更详细说明见文献[16].

KenGen$()\rightarrow(pk,sk)$
1. 生成公钥 $pk=(N,e,g)$和私钥 $sk=(N,d,g)$.
2. 输出(pk,sk).

Add$(usk,F,\alpha,GID)\rightarrow(F',M,\alpha')$
1. 令$(N,d,g)=usk,(B)=\alpha$.
2. 将文件 F 分割成 t 个数据块,$F[1],\ldots,F[t]$,每个数据块的长度为 L 位.
3. 对于 $1\leqslant i\leqslant t$:
   a) 计算数据块 $F[i]$的同态认证标签:$A[i]=((H(GID)\|B+i)\cdot g^{F[i]})^d \bmod N$.
4. 记录 F 在文件组中的下标范围 $R=(B,B+t)$.
5. 更新文件组最大下标 $B'=B+t$.
6. 输出$(F'=F,M=(A[1],\ldots,A[t],R),\alpha'=(B'))$.

Challenge$(\alpha)\rightarrow chal$
1. 生成随机挑战密钥 k_1 和 k_2: $k_1\xleftarrow{R}\{0,1\}^{\kappa},k_2\xleftarrow{R}\{0,1\}^{\kappa}$.
2. 根据目标安全等级定义将要挑战的数据块的数量 c.
3. 输出 $chal=(c,k_1,k_2)$.

Prove$(upk,chal,F',M,\alpha)\rightarrow P$
1. 令$(N,e,g)=upk,(c,k_1,k_2)=chal,(B)=\alpha$
2. 对于 $1\leqslant j\leqslant c$:
   a) 计算要检测的数据块的虚拟下标值: $i_j=\pi[B]_{k_1}(j)$;
   b) 计算系数: $b_j=e_{k_2}(j)$;
   c) 定位要检测的数据块及它们的同态认证元:
   $$f[i_j]=F'[i'_j]=F'[i_j-F'.R.start],a[i_j]=M..A[i'_j]=M..A[i_j-F'.R.start].$$
3. 计算 $a=a[i_1]^{b_1}\cdot\ldots\cdot a[i_c]^{b_c} \bmod N$,这个值应与 $(H(GID\|i_1)^{b_1}\cdot\ldots\cdot H(GID\|i_c)^{b_c}\cdot g^{b_1f[i_1]+\ldots+b_cf[i_c]})^d \bmod N$ 相等.
4. 计算 $f=b_1f[i_1]+\ldots+b_cf[i_c]$.
5. 输出 $P=(a,f)$.

Verify$(upk,chal,P,\alpha)\rightarrow\{0,1\}$
1. 令$(N,e,g)=upk,(c,k_1,k_2)=chal,(a,f)=P,(B)=\alpha$.
2. 对于 $1\leqslant j\leqslant c$:
   a) 计算 $i_j=\pi[B]_{k_1}(j)$ 和 $b_j=e_{k_2}(j)$
3. 计算 $\tau_1=a^e \bmod N,\tau_2=H(GID\|i_1)^{b_1}\cdot\ldots\cdot H(GID\|i_c)^{b_c}\cdot g^f \bmod N$.
4. 如果 $\tau_1=\tau_2$ 则输出 1,否则输出 0.

Fig.1 Algorithms of MF-PDP scheme

图 1 MF-PDP 方案算法

2 隐式可信第三方审计架构

2.1 安全模型和架构

基于隐式 TTP 的数据持有性审计架构的安全模型中存在 3 个角色,即云存储服务提供者(service storage provider,简称 SSP)、用户(user)、隐式数据持有性审计者(data possession auditor,简称 DPA).SSP 是模型中唯一

的威胁来源,有可能违反服务合同而未能完全履行保持用户数据完整性与持有性的责任.DPA 作为 TTP 能和 SSP 协同完成数据持有性审计而不受他方干扰,不与他方合谋,用户只信任由 DPA 生成的审计结果.

由于用独立于 SSP 的机构作为 TTP 存在难于实现和部署、潜在隐私数据泄露等问题,在我们的审计架构中,隐式 DPA 与云服务器集成在一起,并周期性地与 SSP 交互完成数据持有性审计,生成显篡改审计日志.关于某个文件组的所有日志形成日志链表存放在云端,DPA 对审计日志和链表进行签名认证,任何一方都能验证日志及链表的完整性和正确性,实现公开审计.用户在多数时间里可以离线,而只在需要时查看并验证日志来获知数据的完整性.用户、SSP 和 DPA 的交互过程如图 2 所示(步骤 1 和步骤 2 也需要 SSP 和 DPA 交互,为清晰起见,在图中未画出;步骤 2 和步骤 3.1~步骤 3.3 的审计交互过程在一个文件组的生命周期中均可多次执行且可交错进行).需要指出:与传统 TTP 审计不同,隐式 DPA 为被动响应方,每次持有性审计请求由 SSP 发起,且 DPA 生成的审计结果由 SSP 保存.这样最大限度降低了 DPA 的处理和存储需求,使基于可信硬件的实现成为可能.隐式 TTP 审计架构有效降低了 TTP 的部署和运营成本,避免了审计者窃取用户隐私数据的问题.

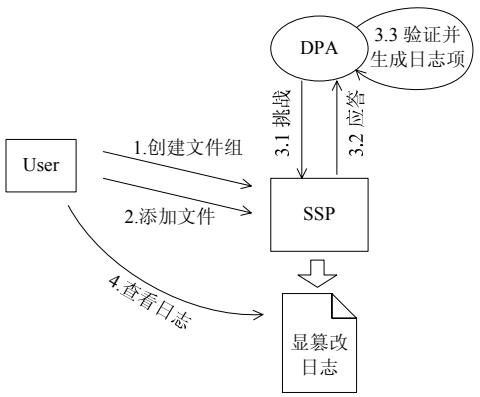


Fig.2 Reciprocal process with implicit third party

图 2 隐式第三方审计架构的三方交互过程

2.2 DPA的实现

我们提出用具有下列特性的防篡改(tamper-resistant)可信硬件实现隐式 DPA:其他实体不能更改其内部程序和状态,不能获取其密钥信息;当检测到针对自己的物理攻击时,DPA 应能对秘密信息执行自我销毁.现有的安全协处理器产品如 IBM 4764[17],可以满足 DPA 的上述功能要求.由于 DPA 部署在云端,SSP 应保证 DPA 的正常工作,不受到来自外界的攻击.因此,不论是因 DPA 自我销毁还是用户数据被损坏造成的日志链表不完整,都判定为 SSP 没有履行相关职责.

在实际云存储环境中,文件组数随用户数增加而增加.由于 SSP 以文件组为单位与 DPA 进行交互,当文件组数量增加时,一个 DPA 可能无力负担.此时,可采用多 DPA 协同工作的模式,以减轻单个 DPA 工作负担,提高系统可扩展性.持有性审计由分布在 SSP 内网中多台服务器上多个 DPA 组成的集群共同完成,每个 DPA 负责系统中一部分用户所拥有的文件组的持有性审计,各 DPA 独立工作,并使 DPA 的数量随文件组的增加而增加.IDPA-MF-PDP 方案中,DPA 的内部状态和执行功能的简单性,保证了上述多 DPA 集群方案的可行性.多 DPA 的任务调度方法和协同工作协议不在本文的讨论范畴.

2.3 显篡改审计日志

审计日志由 DPA 根据数据持有性检查结果生成,是存储在磁盘介质上的结构化记录(图 3).日志项(log entry,简称 LE)是根据持有性检查结果生成的一个日志项,对应于同一个文件组的所有日志项将形成一个日志链表.用户通过调阅文件的日志链表,即可查看该文件组的持有性审计历史.LE 由 5 个字段组成,其中,*result* 是本次持有性检查的结果,结果为 1 表示文件完好,结果为 0 表示文件受损;*time* 为日志生成的时间,用于保证日志项的新鲜性和不可复制性;*eid* 是日志项在云中的唯一标识;*prev_eid* 为同一文件前一次审计对应的日志项标识,用于形成日志链表;*sig* 是 DPA 利用其私钥 *dsk* 对 *result*,*time*,*eid* 和 *prev_eid* 一起的 RSA 签名.

项引用(entry reference,简称 ER)唯一地对应于一个文件组,由 5 个字段构成,其中,*UID* 为用户的身份标识,*GID* 为该文件组的标识,二者在云中是唯一的;*eid* 是一个日志项标识;*time* 是 ER 被创建或修改的时间戳;*sig* 是 DPA 利用其私钥 *dsk* 对前 4 个字段的 RSA 签名.ER 总是与该文件组的最新一次持有性审计产生的日志项 LE

具有相同的 *eid*.当文件组被创建后,DPA 生成一个与之对应的 ER.没有审计过的文件组,即没有对应的日志项,其 *ER.eid* 为−1,且 *ER.sig* 为空.

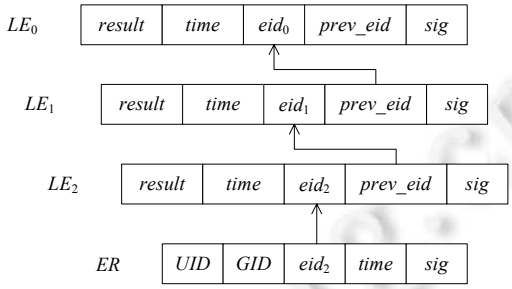


Fig.3 Log list of one file group

图 3 一个文件组的审计日志链

3 三方交互协议

为了将 MF-PDP 与隐式第三方审计架构整合,本节就创建文件组、向文件组添加文件、删除文件组和审计这 4 个操作给出用户、DPA 与 SSP 之间的交互协议.该协议中同时包含了 MF-PDP 的挑战-应答过程以及审计结果的可信生成和保存过程,形成完整的基于隐式 DPA 的多文件数据持有性证明系统(implicit DPA multiple-file PDP,简称 IDPA-MF-PDP).协议的设计原则是:在保证安全性的前提下,尽可能降低对 DPA 的计算和存储需求.下述各协议中假设各方均能获得他方的公钥.

3.1 创建文件组交互协议

图 4 描述了创建文件组的交互过程.假设在协议开始前,DPA 存有所有用户的 *UID*,且 *UID* 在 DPA 和云中都是唯一的.DPA 为每个用户创建的每个文件组维护一个持久状态,保存在α中.

USER→SSP:
  CREATE_GROUIP_REQUEST()
SSP→DPA:
  CREATE_GROUIP_REQUEST(*UID*)
DPA:
  VERIFY_ID(*UID*)
  GID←*UUID*()
  α[*UID*,*GID*]←0
  ER.uid←*UID*
  ER.gid←*GID*
  ER.time←*NULL*
  ER.eid←−1
  ER.sig←*NULL*
  signed_msg←$SIGN\_MSG_{dsk}$(*GID*,α[*UID*,*GID*])
DPA→SSP:
  signed_msg
  ER
SSP:
  write(*UID*,*GID*,α,*ER*)
SSP→USER:
  signed_msg
USER:
  (*GID*,α)←$VERIFY\_MSG_{dpk}$(*signed_msg*)
  write(*GID*,α)

Fig.4 Interaction protocol of creating file group

图 4 创建文件组交互协议

当用户需要创建文件组时,向 SSP 发送创建文件组的请求.SSP 接到请求后,将这个请求连同用户 *UID* 传递给 DPA.DPA 接到请求,完成图 4 中所示工作,并为这个文件组生成一个项引用 ER,置 *ER.time* 与 *ER.sig* 为空,

ER.eid 为−1,表示该文件组尚未被审计.DPA 用自己的私钥 *dsk* 为 *GID* 和α生成签名消息 *signed_msg*,并将其与 ER 返回给 SSP,SSP 从 *signed_msg* 中用提取出的 *GID* 与α,将其与 ER 保存,并将 *signed_msg* 返回给用户.用户需本地保存 *GID* 和状态α.

3.2 添加文件交互协议

图 5 描述了向文件组添加文件的交互过程.用户调用 $MF\text{-}PDP.Add(usk,F,\alpha,GID)\rightarrow(F',M,\alpha')$,将文件 *F* 分为 *t* 块,为每个文件块生成同态认证元,并更新本地该文件组的持久状态α.用户将文件块集合 *F*′、同态认证元集合 *M*、块数 *t* 以及目标组 *GID* 发送给 SSP,请求向文件组添加文件.SSP 将 *UID*,*GID* 及 *t* 发送给 DPA.DPA 接到请求,完成图 5 所示的工作,最终将修改后的文件组信息生成签名消息返回给 SSP.SSP 接到签名消息后,对 *F*′,*M* 和α进行存储,并将签名消息返回给用户进行比对.

USER:
  $MF\text{-}PDP.Add(usk,F,\alpha,GID)\rightarrow(F',M,\alpha')$
USER→SSP:
  $ADD\_REQUEST(F',M,t,GID)$
SSP→DPA:
  (UID,GID,t)
DPA:
  $VERIFY\_ID(UID,GID)$
  $B=\alpha[UID,GID]$
  $\alpha[UID,GID]=B+t$
  $signed\_msg\leftarrow SIGN\_MSG_{dsk}(GID,\alpha[UID,GID])$
DPA→SSP:
  $signed\_msg$
SSP:
  $write(F',M,\alpha')$
SSP→USER:
  $signed\_msg$
USER:
  $(GID,\alpha)\leftarrow VERIFY\_MSG(signed\_msg)$
  $check(GID,\alpha)$

Fig.5 Interaction protocol of adding a file to a file group

图 5 添加文件交互协议

3.3 删除文件组交互协议

图 6 列出了 DPA 与 SSP 交互完成删除文件组操作的过程.

USER→SSP:
  $DELETE\_REQUEST(GID)$
SSP→DPA:
  $GROUIP\_DELETE\_REQUEST(UID,GID)$
DPA:
  $VERIFY\_ID(UID,GID)$
  $DELETE(UID,GID)$
  $signed\_msg\leftarrow SIGN\_MSG_{dsk}(GID,DELETED)$
DPA→SSP:
  $signed\_msg$
SSP:
  $DELETE\_GROUIP(GID)$
SSP→USER:
  $signed\_msg$
USER:
  $(GID,DELETED)\leftarrow VERIFY\_MSG(signed\_msg)$
  $check(GID,DELETED)$

Fig.6 Interaction protocol of deleting a file group

图 6 删除文件组交互协议

用户向 SSP 发起删除文件组的请求,SSP 将这个请求传递给 DPA,DPA 将自身存储的该文件组的记录删除.删除成功后,将 *GID* 及删除标识生成签名消息返回给 SSP.SSP 将该组的文件块、认证元及其他信息全部删除,

并向用户返回签名消息.

3.4 审计交互协议

图 7 描述了 SSP 与 DPA 交互完成文件组审计的过程($SIGN_k(M)$函数用私钥 k 计算消息 M 的签名).SSP 以文件组为单位,周期性地向 DPA 发送 DPC 请求,每个周期用 *epoch* 表示.DPA 收到审计请求后,以该文件组的持久状态α为参数,向 SSP 发起挑战.SSP 根据收到的挑战计算证据 P,并将其连同该文件组的 ER 一起发送给 DPA. DPA 收到证据和 ER 后,验证 ER 的新鲜性和真伪(通过 *time* 和 *sig* 域),验证通过,则生成新的日志项 LE,将对证据的验证结果写入其中,并更新 ER 的 *time* 和 *sig* 域.若此文件为第 1 次接受审计,由图 4 中创建文件组交互协议可知:此时,*ER.eid* 为 *NULL*,因此对应于第 1 次审计结果的日志项 *LE.prev_id* 为 *NULL*.最终,DPA 将 LE 连同 ER 返回给 SSP,由其写入审计日志.

SSP→DPA:
  AUDIT_REQUEST(*UID*,*GID*)
DPA:
  VERIFY_ID(*UID*,*GID*)
  chal←*MF-PDP.Challenge*(*α*[*UID*,*GID*])
DPA→SSP:*chal*
SSP:
  MF-PDP.Prove(*upk*,*chal*,*F'*,*M*,*α*)→*P*
SSP→DPA:(*P*,*ER*)
DPA:
  VERIFY_ER(*ER.time*,*ER.sig*)
  result←*MF-PDP.Verify*(*pk*,*chal*,*P*,*α*[*UID*,*GID*])
  LE.result←*result*
  LE.time←*GET_CUR_TIME*()
  LE.eid←*GET_EID*()
  LE.prev_eid←*ER.eid*
  LE.sig←*SIGN*(*LE.result*,*LE.time*,*LE.eid*,*LE.prev_eid*)
  ER.eid←*LE.eid*
  ER.time←*LE.time*
  ER.sig←*SIGN*(*ER.uid*,*ER.gid*,*ER.eid*,*ER.time*)
DPA→SSP:(*ER*,*LE*)
SSP:*write*(*ER*,*LE*)

Fig.7 Interaction protocol of auditing

图 7 审计过程交互协议

4 性能与安全性分析

4.1 开销复杂度分析

我们从 I/O 开销、计算开销、通信开销和 DPA 存储开销这 4 个方面来分析 IDPA-MF-PDP 的性能.服务器端的磁盘 I/O 开销由读取挑战指定的 c 个抽样块带来.根据文献[16]中的分析,为了以一定的置信度检测一定比例的块损坏发生,所需抽样块个数 c 为常数,与文件组内文件个数 n 无关.

计算开销主要由指数运算引起.在 IDPA-MF-PDP 方案中:

- 服务器在 Prove 阶段需要计算 $a=a[i_1]^{b_1}\cdot\ldots\cdot a[i_c]^{b_c} \bmod N$,其中包含 c 次指数运算;
- DPA 在 Verify 阶段需要计算$\tau_1=a^e \bmod N$ 和 $\tau_2=H(GID\|i_1)^{b_1}\cdot\ldots\cdot H(GID\|i_c)^{b_c}\cdot g^f \bmod N$, 其中包含了 c+2 次指数运算;DPA 在生成日志时需要验证服务器提供的 ER 的签名,并生成新的 ER 和 LE 的签名,其中包含 3 次指数运算.

通信开销由 Challenge 阶段在服务器和 DPA 之间传递的比特数决定.DPA 向服务器端发送挑战 *chal*=(c,k_1, k_2),这个挑战的长度为 $\log_2 T_{\max}$+2κ比特.服务器端向 DPA 返回验证结果 P=(a,f),长度为 L'+$\log_2 N$ 比特(其中,L'是 $f=b_1f[i_1]+\ldots+b_cf[i_c]$的长度,仅比文件块的长度 L 略长).

DPA 存储开销取决于密钥和持久状态的大小.在 IDPA-MF-PDP 方案中:DPA 存储生成和验证同态认证元的

RSA 密钥对(n,e,d),大小为 $2\log_2 N$ 比特;日志项的签名密钥对,大小为 $2\log_2 N$ 比特;持久状态α=(B),大小为 $m\cdot\log_2 T_{\max}$ 比特(其中,m 为此 DPA 管理的文件组个数).

为了比较 IDPA-MF-PDP 的性能,我们引入了两个基本的 MF-PDP 方案作为比较基准,其中,PDP 方案用文献[2]中的方案分别为文件组中每个文件做数据持有性检查;DPDP 方案以文献[4]中第 3.2 节的方案为基础,用一个多层级跳跃表(skip list)维护文件组中所有文件.DPDP 以文件为单位,将文件块的 *tag* 值作为叶子节点计算一个跳跃表,并将文件组中每个文件跳跃表的起始节点(start node)作为叶子节点构造父级跳跃表.假设文件组中有 n 个文件,且这些文件中单个文件块数最大值为 $T_{\max}$,则文件组跳跃表高度为 $\log n+\log T_{\max}$.在 Prove 阶段,DPDP 返回 c 个抽样块的 *tag* 值和文件块聚合值 M 及这些块在跳跃表中的验证路径(verification path);Verify 阶段,首先根据 DPA 本地保存的文件组起始节点验证 c 个 *tag* 值及其返回路径,再做 c 个 *tag* 的乘法聚合值,将其与 M 做比较.3 个方案均基于本文提出的由 DPA 担任隐式第三方的审计架构.

表 1 总结了 IDPA-MF-PDP 的复杂度和开销构成,并且将其与 DPDP 和 PDP 方案做出比较.从表中可以得出:IDPA-MF-PDP 将所有开销均减小到 $O(1)$,其中,DPA 计算开销和存储开销的大幅降低使基于可信硬件的 DPA 得以实现.需要特别指出:DPDP 方案的服务器计算开销虽然不包含指数运算,在表中表示为 $O(1)$,但其需为每个挑战块利用哈希计算验证路径,耗时为 $O(\log n)$.

Table 1 Complexity of PDP, DPDP and IDPA-MF-PDP

表 1 PDP,DPDP 和 IDPA-MF-PDP 审计交互的开销复杂度与构成

方案	服务器 I/O (抽样块个数)	服务器计算开销 (指数运算次数)	DPA 计算开销 (指数运算次数)	服务器与 DPA 间通信开销(比特)	DPA 存储开销(比特)
PDP 复杂度	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
DPDP 复杂度	$O(1)$	$O(1)$	$O(1)$	$O(\log n)$	$O(1)$
IDPA-MF-PDP 复杂度	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
IDPA-MF-PDP 开销构成	c	c	c+5	$\log_2 T_{\max}+2\kappa+L'+\log_2 N+2\lvert ER\rvert+\lvert LE\rvert$	$4\log_2 N+\log_2 T_{\max}$

说明:n 为文件组中文件的个数(在审计交互协议中,服务器端向 DPA 返回响应时,一并将文件组的项引用 ER 返回给 DPA,其大小为$|ER|$.DPA 生成日志,并将其返回给服务器存储时,也会产生常数级的通信开销,包括新的项引用 ER 新生成日志项 LE,其大小为$|ER|+|LE|$).

4.2 安全性分析

我们通过 4 个定理将 IDPA-MF-PDP 的数据持有性审计过程分解为结果的生成(定理 1、定理 2)和结果的保存(定理 3、定理 4)两个环节论证其安全性.我们假设在 IDPA-MF-PDP 架构中 DPA 是可信的,即,外界不能修改 DPA 用于生成日志的时钟、密钥、程序等内部状态和数据(见第 2.2 节).

定理 1. 在 DPA 持有正确的文件组持久状态的前提下,MF-PDP 具有和基本 PDP 方案[2]等同的安全性.

证明:在 MF-PDP 中,我们把文件组作为数据持有性审计的单元.文件组中包含不定数目的文件,每个文件都被分成特定长度的数据块,每个数据块都被赋予一个在文件组中的虚拟下标.在 Prove 阶段,SSP 先根据文件组持久状态为挑战生成一组在文件组中的虚拟下标,然后,通过这组虚拟下标生成同态认证元,计算需要返回的证据 P.同样,在 Verify 阶段,DPA 也需先计算出挑战块的虚拟下标,再将自己计算的验证值与 SSP 返回的证据 P 相比较(见第 1.2 节).这样,一个文件组就可以被看作一个虚拟的大文件,对文件组中任意文件数据块的损坏等价于对虚拟文件数据块的损坏,对文件组内所有文件的审计就归约为对这个虚拟文件的审计,MF-PDP 的安全性就可以规约为单文件数据持有性的安全性.而在文献[2]中,Ateniese 等人已给出严格的单文件 PDP 的安全性证明,本文将不再赘述(关于 MF-PDP 安全性更详尽的分析,参见文献[16]). □

定理 2. 在一个文件组的由正常操作组成的生命周期内,DPA 始终持有正确的该文件组持久状态.

证明:在 IDPA-MF-PDP 中,DPA 为每个文件组保持一个持久状态,作为生成挑战的依据.由于 DPA 是可信的,外界只能通过 SSP 和 DPA 间的交互协议.在创建文件组时,DPA 将文件组的α初始化为 0,保证其初始状态的正确性(第 3.1 节).在向文件组添加文件时,用户与 DPA 都将自身存储的α值进行修改,其中,用户修改后的α值一定

是正确的.DPA 将修改后的α值用自己的私钥进行签名,传递给用户进行比对.用户用 DPA 的公钥对签名消息进行验证并提取出签名消息中的α与自身的α进行比对,如果结果一致,则 DPA 中现有的α也是正确的(第 3.2 节).因此,只要文件组的创建过程以及向文件组添加文件的过程正确执行,则 DPA 中保存的α一直是正确的. □

定理 1 和定理 2 保证了每次数据持有性审计生成结果的可信性.为了给出关于记录审计结果的审计历史的可信性的定理 3 和定理 4,下面先给出 3 个相关定义.

定义 1(项引用 ER 的可验证性). 称项引用 ER 是可验证的,当且仅当:

(1) 用 DPA 的公钥 *dpk* 可验证 *ER.sig*;

(2) *ER.time* 落在距离当前时刻最近的 *epoch* 内.

定义 2(日志项 LE 的可验证性). 称日志项 LE 是可验证的,当且仅当用 DPA 公钥 *dpk* 可验证 *LE.sig*.

定义 3(审计历史的一致性). 称文件组 *G* 的审计历史 *h* 处于一致状态,当且仅当:

(1) 项引用 ER 是可验证的;

(2) *ER.eid* 指向的日志项 LE 存在且可验证;

(3) 对于日志链表中每一日志项 LE,若 *LE.prev_id* 不为空,其指向的 LE 存在且可验证.

定理 3. 当 SSP 正确按照第 3.4 节的交互协议与 DPA 进行交互时,所生成的关于文件组 *G* 的审计历史 *h* 一定处于一致状态.

证明:根据第 3.4 节中的审计过程交互协议,当 DPA 收到来自 SSP 的响应 *P* 及当前文件组的项引用 ER 时,会先验证 *ER.sig* 的正确性和 *ER.time* 的新鲜性,验证通过后方可继续执行.DPA 周期性地发起挑战,在每个 *epoch* 内至少发生一次 ER 的更新,只要 *ER.time* 落在最近的 *epoch* 内,则此 ER 确实为在最新一次审计过程中修改过的 ER.DPA 生成用其私钥签名的保存本次审计结果的 LE,并生成新的 ER 使其 eid 指向 LE.根据定义 1~定义 3 易知:若 SSP 正确按照交互协议与 DPA 进行交互,得到的审计历史 *h* 一定处于一致状态.

用户可按照如下过程验证某个文件组 *G* 的审计历史 *h* 的一致状态:向 SSP 发送文件组 GID,SSP 根据用户请求返回该文件组的日志链表 *l*;用户接收到 *l* 后首先验证 ER,接着逐项验证链表中日志项 LE,直至最后一个可验证的 LE 的 *prev_id* 域为空为止. □

定理 4. 文件组 *G* 的审计历史 *h* 在某个 *epoch* 内处于一致状态 s_1,则在 *h* 达到下一个 *epoch* 内的一致状态 s_2 之前,任何针对 *h* 一致性的攻击(修改或丢弃)都无法在 DPA 与 SSP 审计交互过程中通过交互协议,或者无法在用户审查 *h* 时正确通过用户审计,而使 *h* 仍处于一致状态.

证明:假设 *h* 在一致状态 s_1 时的索引项为 ER_1,敌手在 *h* 到达下一个一致状态 s_2 前试图对其篡改.下面分 3 种情况讨论:(1) 如果敌手试图修改 ER_1,由于敌手无法掌握 DPA 的私钥,则其不能伪造 *ER.sig*,所以其对 ER 的任何字段的修改将违背定义 3 的条件(1),使审计历史 *h* 偏离一致状态;(2) 如果审计日志链表不空且敌手试图修改或丢弃第一个 LE,将违背定义 3 的条件(2),*h* 偏离一致状态;(3) 假设审计日志链表日志项个数大于 1 且敌手试图修改或丢弃除第一个 LE 外的其他 LE,将违背定义 3 的条件(3),*h* 偏离一致状态.由以上分析可知:对审计历史 *h* 的任何部分的修改或丢弃,都会使其偏离在某个 *epoch* 内的一致状态. □

定理 3 和定理 4 保证了 SSP 和 DPA 审计交互过程中生成处于一致状态的审计历史,且该历史一旦生成就无法篡改.综上,IDPA-MF-PDP 的数据持有性审计是安全的.

5 系统实现和实验评估

5.1 系统实现

我们在 Linux 平台上用 C 语言实现了 IDPA-MF-PDP 原型系统,同时实现了第 4.1 节中提到的适用于文件组的 DPDP 方案以及基于单文件 PDP 方案[2]的原始 MF_PDP 方案作为对照,在后文中,分别用 DPDP 和 PDP 表示.所有密码操作实现均来自于 OpenSSL(版本号 0.9.8o)[18].我们采用两台配置相同的服务器来模拟云服务器和 DPA:Intel Xeon 四核处理器,8G 内存(主频 2.27GHz),10000r/m 146G SAS 硬盘.由于可信硬件运算速度一般比云服务器慢,我们根据对测试平台测试获得的数据加入特定的减速因子,以模拟可信硬件的性能.IBM4764 每秒钟

能产生 2.16 个长度为 1 024 比特的 RSA 密钥对[17].对比在测试平台获得的数据(14.92 个/s),我们设定可信硬件对大数运算的减速因子为 6.91.

测试文件大小在区间[0.5GB,1GB]内均匀分布,分块大小为 4KB.为保证 DPDP 方案中文件组跳跃表层数最少以达到最高性能,我们将 DPDP 方案中各文件大小设为一致.选择每次挑战块个数 c=460,使得以 99%的置信度检测 1%的块损坏比例.

5.2 实验结果

我们测试了 IDPA-MF-PDP,DPDP 和 PDP 方案的审计性能.一次审计交互时间由 4 部分组成:云服务器读取挑战数据块的 I/O 时间、云服务器计算持有性证据的计算时间、DPA 验证证据的时间和 DPA 与服务器的通信时间.由于在我们的隐式第三方审计架构中 DPA 整合在云端,DPA 和云服务器通信属于本地通信,且传输数据量为常数量级,其时间开销可忽略不计.

图 8 给出了 IDPA-MF-PDP,DPDP 和 PDP 的 I/O 时间开销.由于 IDPA-MF-PDP 与 DPDP 的 I/O 内容几乎一致,且两个方案在挑战过程中选取的文件块数量均不随文件组大小变化,因此在图 8 中将两方案合并.由图 8 可知:当文件组中的文件数目增加时,IDPA-MF-PDP/DPDP 的 I/O 时间基本维持不变.

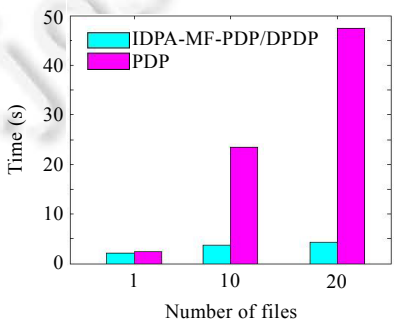


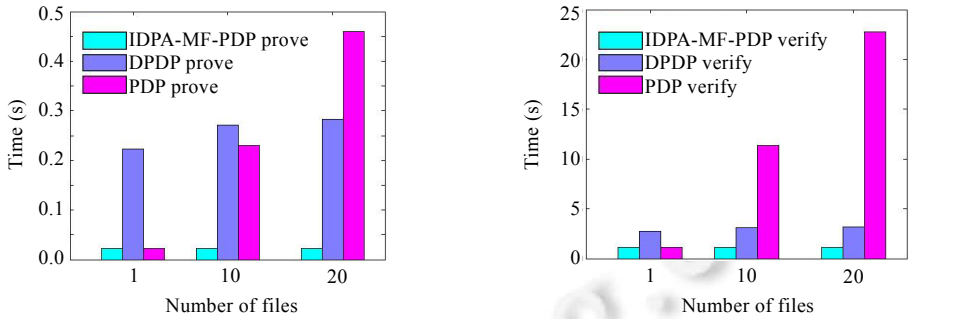
Fig.8 I/O overhead of IDPA-MF-PDP/DPDP and PDP

图 8 IDPA-MF-PDP 和 PDP 的 I/O 开销

图 9(a)给出了 IDPA-MF-PDP,DPDP 和 PDP 方案在 Prove 阶段的计算时间.IDPA-MF-PDP 中,挑战块的数目不随文件组大小而更改,因此计算时间也不会增长;DPDP 挑战块数目虽然恒定,但 Prove 阶段需根据跳跃表为每个挑战块计算验证路径,其耗时随文件块数目增长呈对数增长;而在 PDP 中,计算时间会随着挑战块数的线性增加而显著增长.

图 9(b)给出了 3 个方案 Verify 过程的计算时间.由于此过程由 DPA 验证,因此我们将所有数据乘以减速因子.由图可知:即使 Verify 阶段整体上比 Prove 阶段耗时久,但是随着文件数目的增加,IDPA-MF-PDP 的计算速度速度基本保持恒定,DPDP 仍因维护跳跃表使耗时呈对数增加,PDP 计算速度随文件数目增长最快(在 Verify 过程中,虽然 DPA 做了生成日志的工作,但测试结果表明,生成日志的时间基本恒定在 3ms 左右,对 Verify 时间影响甚微,因此忽略不计).结合图 8 和图 9 可以看出,I/O 开销占 IDPA-MF-PDP 审计开销的主要部分.

表 2 给出当文件组中文件数目为 100 时,IDPA-MF-PDP,DPDP 和 PDP 总审计时间对比及其组成.为了计算通信开销,我们将 SSP 与 DPA 之间的带宽设为 100Mbps.IDPA-MF-PDP 的通信内容为一个文件块的聚合值及一个认证元的聚合值,易知,PDP 的通信内容是其 100 倍.根据文献[4],DPDP 的 Prove 阶段,SSP 返回给 DPA 的持有性证据包括 c 个文件块 tag 值、c 个验证路径以及一个文件块聚合值,其中,每个验证路径包括 $\log n+\log T_{\max}$ 个哈希值.除此之外,3 个方案的通信内容均包含日志相关内容,其中,PDP 日志开销为其余两个方案的 100 倍.从表中可以看出:当文件数目增加时,IDPA-MF-PDP 的审计时间和计算时间均保持在常量水平,而 DPDP 和 PDP 的审计时间和计算时间均有明显增加.其中,DPDP 传输验证路径哈希值的通信开销多至 23.4s,导致其总开销显著高于 IDPA-MF-PDP.



(a) IDPA-MF-PDP, DPDP 和 PDP 在 Prove 阶段耗时比较 (b) IDPA-MF-PDP, DPDP 和 PDP 在 Verify 阶段耗时比较

Fig.9 Computational overhead in phase Prove, Verify of IDPA-MF-PDP,DPDP and PDP

图 9 IDPA-MF-PDP,DPDP 和 PDP 在 Prove 和 Verify 阶段耗时比较

Table 2 Overheads of IDPA-MF-PDP, DPDP and PDP

表 2 IDPA-MF-PDP,DPDP 和 PDP 的时间开销

时间开销	IDPA-MF-PDP (s)	DPDP (s)	PDP (s)
I/O	4.708 (80.15%)	4.6	235.5
Prove	0.023 (0.39%)	0.307	2.3
Verify	1.14 (19.41%)	3.356	114.015
Communication	0.000 3 (0%)	23.388	0.032
Log Generation	0.003 (0.05%)	0.003	0.3
Total	5.874	31.654	352.147

6 相关工作

数据持有性和可取回性证明,近年来作为云存储安全中一个重要问题[19,20]已得到广泛关注.Ateniese 等人[2]和 Juels,Kaliski[21]首先提出了数据持有性证明和可取回性证明的定义和方案.Ateniese 等人[2]定义了 PDP 模型,并且给出了更高效的实现方案——E-PDP.在他们的方案中,通过随机抽样来减少检查的开销.基于 RSA 的同态认证标签用来提供抽样块的集合值用于验证,因此,此方案支持公开验证.同态认证元同样也是我们方案中的一个重要组成部分.

Ateniese 等人的原始 PDP 方案只支持静态文件.在他们的后续工作中,Ateniese 等人提出了一个动态的 PDP 方案[3].其基本思想是:在系统建立阶段,提出所有后续可能的挑战,并且把预计算的持有性证据当作元数据存储.这样,验证端发起的挑战次数是限定的.此外,每一次对于数据的更新都需要重新计算所有的挑战,这在大文件的存储中会出现问题.Wang 等人[5]研究了分布系统中的动态数据存储问题,他们的方案不仅可以判定数据的正确性,还可以定位数据出错的位置.在他的另一个研究中[6],构造了一个整合了 TTP 公开审计和动态数据的方案.全动态数据更新,特别是数据块的插入,通过用默克尔哈希树存储认证标签来实现.Erway 等人[4]对 Ateniese 提出的 PDP 模型[2]进行扩展,分别利用跳跃表(skip list)和 RSA 树构建了基于秩次的认证字典,以此提出了支持全动态更新的 PDP 方案.但是这个方案的更新操作的效率同样较低.Li 等人[7]提出了 *SN-BN* 表结构来支持数据块的插入操作,他们采用这个表将数据块的逻辑下标和实际位置对应起来.在 PDP 过程中,通过序列号(*SN*)标识挑战的数据块,通过块号(*BN*)取得这些数据块.当插入一个新数据块时,按照当前数据块的数目,顺序设定这个数据块的 *SN*,以此来完成数据的更新.

以上支持动态数据的 PDP 方案[3−7]都针对一般的单文件更新,且方案复杂更新和检查开销较大,尤其是在简单扩展到处理多文件时($O(n)$).其中,只有文献[4]考虑了多文件检查问题,他们将多个可更新文件的组成和目录结构相同的树结构.由于针对最一般的文件系统更新模式,使更新和检查复杂度为 $O(d\cdot\log_2 n)$,其中,d 为树的深度,n 为跳跃表中最大叶节点个数.本文针对云存储中数据更新的特殊模式提出的 MF-PDP 方案,在保持方案

简单性的同时,显著降低了多文件检查的开销($O(1)$).

在 Wang 等人工作[6]的基础上,一些学者考虑了基于第三方审计的 PDP 方案的在实际云存储环境中部署所面临的问题和解决方案,主要集中在多 SSP 和可扩展性两个方面.朱岩等人[8]考虑了多个 SSP 协同提供云存储服务的情形,基于多证明者零知识证明系统构造了协同 PDP 方案.杨侃等人[9]在支持动态审计的同时,通过多用户和多 SSP 的批量审计进一步提高效率.该思想和本文的 MF-PDP 类似,但他们并没有考虑多文件批量审计问题.王化群等人[10]构造了多 SSP 环境下的基于身份的分布式 PDP 方案,提高系统的可扩展性.

这些方案都假设一个 TTP 的存在,而没有关注怎样以可操作的方式实现和部署 TTP,也没有探讨在用户离线的情况下如何将 TTP 的审计结果返回给用户的问题.与这些工作不同,本文并未给出某个具体基于第三方审计的 PDP 方案,而是提出一个一般的隐式 TTP 架构,用部署在云端的可信硬件实现隐式 DPA,通过 DPA 和 SSP 的交互和显篡改审计日志支持用户离线审计.需要指出:该架构具有和多种已有 PDP 方案集成的能力,本文的 IDPA-MF-PDP 即显示了该架构和一种特定的 PDP 方案——MF-PDP 的集成.

7 总 结

本文分析了现有 PDP 方案在真实云存储环境中应用存在的问题,并给出了一个解决方案:基于隐式第三方审计框架的多文件数据持有性证明系统 IDPA-MF-PDP.理论分析表明:IDPA-MF-PDP 具有和单文件 PDP 方案等同的安全性,审计日志提供了可信的审计结果历史记录,IDPA-MF-PDP 是安全的.复杂度分析和实验结果表明:持有性审计的计算和通信开销由线性减少到接近常数水平,其性能主要受限于硬盘 I/O 能力,而非密码运算,IDPA-MF-PDP 是高效的.本文工作为最终实现可在真实云存储环境中部署的数据持有性证明提供了可能.

References:

[1] Feng DG, Zhang M, Zhang Y, Xu Z. Study on cloud computing security. Ruan Jian Xue Bao/Journal of Software, 2011,22(1): 71−83 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3958.htm [doi: 10.3724/S.P.J.1001.2011.03958]

[2] Ateniese G, Burns R, Curtmola R, Herring J. Provable data possession at untrusted stores. In: Ning P, De Capitani di Vimercati S, Syverson PF, eds. Proc. of the ACM Conf. on Computer and Communications Security. New York: ACM Press, 2007. 598−609. [doi: 10.1145/1315245.1315318]

[3] Ateniese G, Pietro RD, Mancini LV, Tsudik G. Scalable and efficient provable data possession. In: Proc. of the 4th Int'l Conf. on Security and Privacy in Communication Networks. New York: ACM Press, 2008. [doi: 10.1145/1460877.1460889]

[4] Erway C, Kupcu A, Papamanthou C, Tamassia R. Dynamic provable data possession. In: Al-Shaer E, Jha S, Keromytis AD, eds. Proc. of the ACM Conf. on Computer and Communications Security. New York: ACM Press, 2009. 213−222. [doi: 10.1145/1653662.1653688]

[5] Wang C, Wang Q, Ren K, Lou W. Ensuring data storage security in cloud computing. In: Proc. of the 2009 IEEE 17th Int'l Workshop on Quality of Service (IWQoS 2009). Piscataway: IEEE, 2009. 1−9. [doi: 10.1109/IWQoS.2009.5201385]

[6] Wang Q, Wang C, Li J, Ren K, Lou W. Enabling public verifiability and data dynamics for storage security in cloud computing. In: Backes M, Ning P, eds. Proc. of the14th European Symp. on Research in Computer Security (ESORICS 2009). Saint-Malo, Berlin: Springer-Verlag. 2009. 355−370. [doi: 10.1007/978-3-642-04444-1_22]

[7] Li C, Chen Y, Tan P, Yang G. An efficient provable data possession scheme with data dynamics. In: Proc. of the 2012 Int'l Conf. on Computer Science and Service System. Piscataway: IEEE, 2012. 706−710. [doi: 10.1109/CSSS.2012.182]

[8] Zhu Y, Hu H, Ahn H, Yu M. Cooperative provable data possession for integrity verification in multi-cloud storage. IEEE Trans. on Parallel and Distributed Systems, 2012,23(12):2231−2244. [doi: 10.1109/TPDS.2012.66]

[9] Yang K, Jia XH. An efficient and secure dynamic auditing protocol for data storage in cloud computing. IEEE Trans. on Parallel and Distributed Systems, 2013,24(9):1717−1726. [doi: 10.1109/TPDS.2012.278]

[10] Wang H. Identity-Based distributed provable data possession in multi-cloud storage. Trans. on Services Computing, 2014,PP(99):1. [doi: 10.1109/TSC.2014.1]

[11] Internet and cloud services—Statistics on the use by individuals. http://ec.europa.eu/eurostat/statistics-explained/index.php/Internet_and_cloud_services_-_statistics_on_the_use_by_individuals

[12] Librsync. https://github.com/dropbox/librsync

[13] Tarsnap. http://www.tarsnap.com/

[14] Vrable M, Savage S, Voelker GM. Cumulus: Filesystem backup to the cloud. ACM Trans. on Storage (TOS), 2009,5(4):1−28. [doi: 10.1145/1629080.1629084]

[15] Vrable M, Savage S, Voelker GM. BlueSky: A cloud-backed file system for the enterprise. In: Proc. of the 10th USENIX Conf. on File and Storage Technologies (FAST 2012). 2012.

[16] Xiao D, Yang Y, Yao W, Wu C, Liu J, Yang Y. Multiple-File remote data checking for cloud storage. Computers & Security, 2012, 31(2):192−205. [doi: 10.1016/j.cose.2011.12.005]

[17] IBM 4764 PCI-X cryptographic coprocessor. http://www-03.ibm.com/security/cryptocards/pcixcc/overperformance.shtml

[18] Openssl Crypto Library. http://www.openssl.org/

[19] Chen LX, Xu L. Research on provable data possession and recovery technology in cloud storage. Journal of Computer Research and Development, 2012,49(Suppl.):19−25 (in Chinese with English abstract).

[20] Fu YX, Luo SM, Shu JW. Survey of secure cloud storage system and key technologies. Journal of Computer Research and Development, 2013,50(1):136−145 (in Chinese with English abstract).

[21] Juels A, Kaliski B. PORs: Proofs of retrievability for large files. In: Ning P, De Capitani di Vimercati S, Syverson PF, eds. Proc. of the ACM Conf. on Computer and Communications Security. New York: ACM Press, 2007. 584−597. [doi: 10.1145/1315245.1315317]

附中文参考文献:

[1] 冯登国,张敏,张妍,徐震.云计算安全研究.软件学报,2011,22(1):71−83. http://www.jos.org.cn/1000-9825/3958.htm [doi: 10.3724/S.P.J.1001.2011.03958]

[19] 陈兰香,许力.云存储服务中可证明数据持有及恢复技术研究.计算机研究与发展,2012,49(Suppl.):19−25.

[20] 傅颖勋,罗圣美,舒继武.安全云存储系统与关键技术综述.计算机研究与发展,2013,50(1):136−145.

肖达(1981－),男,黑龙江哈尔滨人,博士,讲师,主要研究领域为云存储安全,存储系统.

孙斌(1967－),女,博士,副教授,主要研究领域为计算机网络,信任管理.

杨绿茵(1991－),女,硕士生,主要研究领域为云存储安全.

郑世慧(1979－),女,博士,讲师,主要研究领域为密码学,密码方案的分析与设计.