云存储中支持失效文件快速查询的批量审计方法

王惠峰 李战怀 张 晓 孙 鉴 赵晓南
(西北工业大学计算机学院 西安 710072)

摘 要 云存储服务中,批量审计是高效验证云端数据完整性的关键技术.批量审计容易遭受"失效文件"攻击,并且查询失效文件代价高、速度慢,严重影响着批量审计方案的可用性和效率.针对该问题,提出一种支持失效文件快速查询的批量审计方法,该方法通过建立批量审计过程的关联性,改变了二分查询树中右孩子节点的计算方式,减少了整个查找过程的批量审计次数;并在批量审计过程中执行幂指测试,通过一次审计就可完成含有单个失效文件的子树查找过程,有效缩短了子树的查找长度;采用混合型查询方法,根据历史查询信息设置幂指测试的深度,降低了"失效文件聚集处"的查询开销.安全分析和性能表明,该方法能够快速完成失效文件定位,有效抵抗"失效文件"攻击,保证了批量审计方案的可用性和效率.在少量文件失效的情景下,相较于简单二分查找方法,文中方法耗费的批量审计次数减少了 30%.

关键词 数据安全;云存储;数据完整性验证;批量审计;快速查询;失效文件
中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2017.02338

Batch Auditing Supporting Fast Searching Invalid Files in Cloud Storage

WANG Hui-Feng LI Zhan-Huai ZHANG Xiao SUN Jian ZHAO Xiao-Nan
(*School of Computer Science, Northwestern Polytechnical University, Xi'an* 710072)

Abstract Batch auditing is the key technology to efficiently verify data integrity in the cloud storage service. However, the batch audit schemes are vulnerable to "invalid files" attacks. Searching invalid files brings heavy cost and the search speed is slow, which seriously affect the availability and efficiency of batch audit. Especially, the system still has to run many batch auditing processes to search the invalid files when there are only a few bad files in them. It is the common phenomenon in commercial cloud storage service, which generally does not appear large area of damaged files since the provider may try their best to avoid the worst case. So we focus mainly on the batch auditing with only a small few of invalid files in the process. To solve this problem, this paper proposes a batch audit scheme with fast searching invalid files (FSBA) in cloud storage service. Through establishing the relationship of batch audit process, this method changes the calculation of right child nodes of binary search trees in order to reduce the number of Batch Verification (BV) in the whole the search process. We can get the results of the right nodes by lightweight computing using the intermediate results instead of running the heavy task of BV. If there is only one invalid file in a batch auditing, it may waste a lot of times by using the binary search method. Because it has to follow the search path to verify the validity of the nodes

收稿日期:2016-12-13;在线出版日期:2017-03-30.本课题得到国家"八六三"高技术研究发展计划项目基金(2013AA01A215)、国家自然科学基金(61472323,61502392)、中央高校基本科研业务费专项资金项目(3102015JSJ0009)、华为创新基金项目(YB2014040023)资助.
王惠峰,男,1986 年生,博士研究生,主要研究方向为海量数据存储、云存储安全. E-mail: wanghuifeng12@163.com. **李战怀**,男,1961 年生,博士,教授,中国计算机学会(CCF)会员,主要研究领域为数据库理论、海量数据存储. **张 晓**(通信作者),男,1978 年生,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为海量数据存储、绿色存储. E-mail: zhangxiao@nwpu.edu.cn. **孙 鉴**,男,1982 年生,博士研究生,主要研究方向为海量数据存储、绿色存储. **赵晓南**,女,1979 年生,博士,讲师,中国计算机学会(CCF)会员,主要研究方向为海量数据存储、分级存储.

until the leaf node, which only contains the invalid file. By executing exponents test in batch audit, our method can finish the search of the sub-tree containing one invalid file only through an audit. It can effectively shorten the sub-tree search length. If there are more than one invalid file in the sub-tree, it can result in the failure of running exponents test. In order to significantly reduce the impact of the side effects, we propose the hybrid method called hybrid binary fast search (H-BFS). According to the historical query information, the hybrid search method set the depth of exponents test to reduce the cost of aggregation parts of invalid files. Based the exponent test depth, H-BFS can timely prevent the auditing system from running the exponents test. And H-BFS also can find the invalid files as soon as possible when the invalid file exists in some sub-tree. In addition to this, we also propose the security design for the scheme. Through adding some random values in the process for generating proofs, our scheme can effectively resist replay attack and forgery attack. The design also can ensure the security of batch audit, which the batch audit is passed if and only if the files are all valid. Security analysis and performance analysis show that our proposed methods can quickly locate the invalid files to efficiently resist the "invalid files" attack. It ensures the availability and efficiency of batch audit scheme. Compared to simple binary search, the number of batch audit spent by our methods is reduced by 30% under the scenario with small invalid files.

Keywords　data security; cloud storage; data integrity checking; batch auditing; fast search; invalid file

1　引　言

云存储服务作为互联网＋应用的重要支撑技术，为用户(企业和个人)提供了低成本、高质量、易扩展的在线数据存储服务.很多互联网＋应用以云存储服务为载体，例如协同办公、安全食品、远程诊疗、智慧城市等，这些应用不仅有助于企业转型，而且将引领社会的变革.与此同时，由于云存储服务面临着许多安全风险，导致用户极度担忧数据的存储安全[1-2].频频发生的数据存储安全事故进一步加大了用户的忧虑.例如，2011 年 3 月，谷歌 Gmail 邮箱出现故障，导致大约 15 万用户的数据丢失[2]；2012 年 8 月，盛大云因物理服务器磁盘损坏造成用户数据丢失.数据存储厂商 EMC 指出，64%的受调查企业在过去 12 个月中经历过数据丢失或宕机事故.对用户而言，文件损坏或者丢失(统称为文件失效)是无法接受的，其破坏了数据完整性.检测和查询云存储服务中的失效文件成为一个重要的研究问题.

近年来，针对失效文件的检测问题，研究人员提出了多种面向云存储服务的数据完整性批量审计方案[3-16].但现有方案大都只着眼于批量审计的功能实现，较少关注批量审计失败后的失效文件查询问题.Ateniese 等人[3]提出的数据持有性证明模型(Provable Data Possession，PDP)，不下载整个文件即可完成远端数据的完整性验证.为了提高审计效率，Wang 等人[9]首先利用双线性对函数的性质提出了支持多任务批量审计(Batch Verification，BV)的 PDP 方案.Zhu 等人[10]提出协作式数据持有性验证 Cooperative PDP，实现了多云环境下单用户文件的批量审计.Yang 等人[11]则进一步提供了多云多用户环境下的批量审计方法.

虽然上述批量审计方案易于检测文件的好坏，但是面临一个共同问题：单个文件失效就能导致整个批量审计失败.而后查询失效文件将带来大量计算开销和通信开销，严重影响了批量审计的效率.因此，解决失效文件查询问题，对增强批量审计方案的可用性和效率具有重要意义.

针对失效文件查询定位问题，Wang 和 Yang 等人[3-6]采用简单二分查找(Simple Binary Search，SBS)方法，分组文件并递归执行批量审计，直到找到失效文件.SBS 算法简单却易遭受"失效文件"攻击，即少量失效文件将导致大量审计.Hwang 等人[14]提出一种基于矩阵的查询方法(Matrix-Based Search，MBS)，将文件以矩阵形式放置，批量审计"同行""同列"文件，从失效行和失效列的交叉部分确定失效文件.MBS 起步代价大且易受文件位置影响，例如，位于矩阵对角线的少量文件就可以导致

MBS 查询失败. 基于幂指测试的方法[16](Exponents Test Based Search,ETS)利用失效文件的验证值的非 1 特性,通过对比带指数和不带指数的验证值来定位失效文件位置. 但 ETS 方法只适用于单个文件失效情景,不能很好应用于实际环境. 文献[17-19]提供的失效数字签名查询方法与本文相似,但其灵活性不高,且不能有效抵抗"失效文件"攻击.

为了解决以上问题,本文提出了一种支持失效文件快速查询的批量审计方法(Batch Auditing with Fast Searching Invalid Tasks,FSBA),该方法不仅能够批量确定文件好坏,而且实现了失效文件的快速查询,可以有效抵抗"失效文件"攻击.

本文的主要贡献如下:

(1)提出了一种面向云存储服务的数据完整性的批量审计模型,通过一次审计即可完成多个文件的完整性验证,同时支持批量审计失败后的失效文件确定.

(2)针对不同应用情境,提出了三种失效文件快速查询方法:二分快速查找方法(Binary Fast Search,BFS),建立了审计结果间的关联性,减少了批量审计次数;基于幂指测试的二分快速查找方法,(Exponents Test Based BFS,ET-BFS)降低了二分查找树的高度;混合型的查询方法(Hybrid BFS,H-BFS)降低了"失效文件聚集处"的查询开销.

(3)通过安全分析和性能分析,表明本文的方法是安全和高效的,不仅能够抵抗重放攻击和伪造攻击,还能够快速完成失效文件定位,抵抗"失效文件"攻击,保证了批量审计方案的可用性和高效性.

2 问题描述和预备知识

2.1 系统模型

数据完整性审计系统由云存储用户、云存储服务器、第三方审计者组成,如图 1 所示.

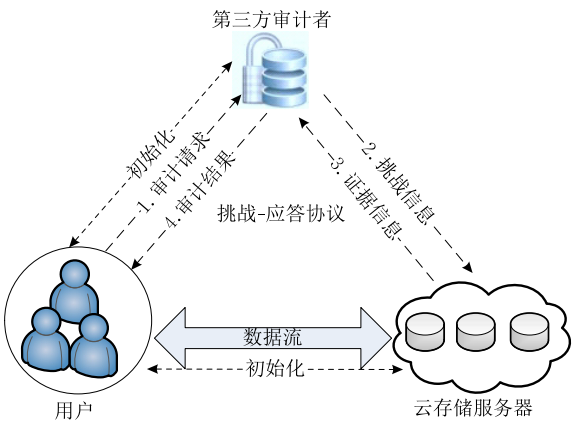

图 1 系统模型

用户(Users,U)(包括个人用户和企业用户)是云存储服务的使用者,同时通过提交数据审计请求验证云端数据的完整性;云存储服务器(Cloud Storage Server,CSS)由云存储服务提供商构建和管理,为用户提供计算和数据存储服务. 云存储服务提供商有义务保证云端数据安全,但是用户对云存储服务提供商的安全承诺存有疑虑;第三方审计者(Third Party Auditor,TPA)是一个具有专业审计知识和能力的独立实体,接收用户的审计请求,代替用户执行文件审计,以减轻用户的审计负担.

U 是用户集合,用户 $U_k(k\in U)$ 具有密钥对 $\{sk_{t,k},pk_{t,k}\}$ 和文件 M_k,将文件 M_k 拆分成 n_k 个数据块,使用自己的密钥对为每个数据块生成认证标签 $t_{i,k}$,构成认证标签集合 $T_k=\{t_{i,k}\mid i\in[1,n_k]\}$,存储 M_k 和 T_k 到云端就完成系统的初始化,后续可以定期执行数据审计.

根据审计方式不同,审计过程分为"单独审计"和"批量审计";单独审计时,审计者按照"One By One"方式,依次响应用户的文件审计请求,完成文件完整性检测;为了提高审计效率,审计者采用"批量审计"方式,同时响应多个用户的审计请求,通过一次审计即可完成多个文件的完整性验证.

批量审计可以极大提升系统的审计效率,但是面临一个严重问题:失效文件将导致整个批量审计失败,而查询失效文件代价大,部分抵消了批量审计的效率优势. 失效文件的位置和数量的不确定性,进一步增大了查询的难度. 例如,文件集合 $M=\{M_1,M_2,\cdots,M_{16}\}$ 中含有若干失效文件 $\{M_2,M_7,M_{11},M_{13},M_{14}\}$,使得批量审计产生的证据集合 $X=\{X_1,X_2,\cdots,X_{16}\}$ 同样存在失效证据,从而导致该批量审计失败;为了确定失效文件位置,需要不断缩小范围,执行多次批量审计. 这不仅增加了审计成本,还严重影响着批量审计系统的可用性. 因此,如何快速查询失效文件成为批量审计必须解决的重要问题.

2.2 安全模型

数据完整性验证模型面临 3 种类型的攻击:

(1)重放攻击和伪造攻击. 重放攻击指 CSS 不执行证据生成过程,仅发送一个已使用的数据持有性证据期待通过审计. 伪造攻击指恶意的 CSS 试图利用已知信息伪造"合法"证据. 为抵抗上述的攻击,本文在挑战信息中加入随机值,在 CSS 返回的数据持有性证据信息与随机值之间建立强依赖关系,使得证据不可伪造和重放.

(2)"失效文件"攻击指 CSS 故意提供少量(一个或者多个)错误的证据信息,造成批量审计失败,

以增大批量审计的代价.面对此类攻击,本文提供两种快速查找方法 ET-BFS 和 H-BFS,执行少量审计即可发现所有的失效文件.

2.3　预备知识

双线性对映射是执行数据完整性验证的基础函数,定义如下:存在两个阶数为 p(p 为大素数)的乘法循环群 G_1 和 G_2,G_1 的生成元为 g.若映射 $e:G_1\times G_1\rightarrow G_2$ 满足如下性质,则称 e 为双线性对映射:

(1)可计算性.存在一个高效的算法可以计算出映射 e;

(2)双线性.对于所有 $u,v\in G_1$ 和 $a,b\in\mathbb{Z}_p$,$e(u^a,u^b)=e(u,v)^{ab}$ 均成立;

(3)非退化性.$e(g,g)\neq 1_{G_2}$,其中 1_{G_2} 表示群 G_2 的单位元.

由双线性对映射的定义可以得到:

(1)$e(u_1\cdot u_2,v)=e(u_1,v)\cdot e(u_2,v)$,$u_1,u_2,v\in G_1$.

(2)$e(a\cdot u,v)=e(u,v)^a$,$u,v\in G_1$,$a\in\mathbb{Z}_p$.

3　支持失效文件快查的批量审计方案

本节首先介绍批量审计模型 FSBA,然后提出三种适应不同应用情境的失效文件快速查询方法.

3.1　FSBA 模型

选取一个双线性对映射 $e:G_1\times G_1\rightarrow G_2$,哈希函数 $h:\{0,1\}^*_{sk}\rightarrow G_1$ 可以将字符串(以 sk 加密)转换为群 G_1 上的元素.

定义 1.　支持失效文件快速查询的批量审计模型 FSBA 由 6 个多项式算法组成:

(1)密钥生成 $keyGen(1^\lambda)\rightarrow(sk_{t,k},pk_{t,k},sk_{h,k})$:$\lambda$ 为安全参数,用户 U_k 选择 2 个随机数 $sk_{t,k},sk_{h,k}\in G_1$,计算 $pk_{t,k}=g^{sk_{t,k}}$,得到用户 U_k 用于生成认证标签 tag 的公私密钥对$(sk_{t,k},pk_{t,k})$和用于加密文件信息哈希密钥 $sk_{h,k}$.

(2)用户 U_k 生成数据块认证标签集合 $TagGen(M_k,sk_{t,k},sk_{h,k})\rightarrow T_k$:输入文件 $M_k=\{m_1,m_2,\cdots,m_n\}$,认证标签私钥 $sk_{t,k}$ 及文件信息加密密钥 $sk_{h,k}$,输出数据块认证标签集合 $T_k=\{t_{i,k}\mid i\in[1,n]\}$,计算 $t_{i,k}$ 如式(1).

$$t_{i,k}=(h(sk_{h,k},FID\|i)\cdot u_k^{m_i})^{sk_{t,k}} \tag{1}$$

FID 是文件标示符,i 是数据块索引,u_k 是随机值.为了简化分析,本文设定每个用户只审计一个文件,故用户序号和审计文件序号含义相同.

(3)审计者生成挑战 $BChall(\{M_{\text{info},k}\}_{k\in Uchal})\rightarrow C$:根据文件信息 $M_{\text{info},k}$ 选取被挑战数据块索引号,并与伴随信息组成挑战信息集合 $C=\{C_k\mid k\in Uchal\}$,其中,$Uchal$ 是被挑战用户集合,集合大小 $|Uchal|$ 为 N.C_k 的内容如式(2)所示,为每个数据块选取随机值 $v_i\in\mathbb{Z}_p^*$,挑战戳 R_k 用于区分不同的挑战,$R_k=(pk_{t,k})^r$,$r\in\mathbb{Z}_p^*$.

$$C_k=(\{i,v_i\}_{i\in Q_k},R_k) \tag{2}$$

(4)云存储服务器生成数据持有性证据信息 $BProve(\{M_k,T_k,C_k\}_{k\in Uchal})\rightarrow P$:服务器收到挑战信息 C_k 后,根据文件信息 M_k 及相应的认证标签 T_k 生成证据信息 P_k,P_k 由标签证据 TP_k 和数据证据 DP_k 组成,计算如式(3)、(4),$P=\{P_k\}_{k\in Uchal}$.

$$TP_k=\prod_{i\in Q_k}t_{i,k}^{v_i} \tag{3}$$

$$DP_k=e(u_k,R_k)^{\sum\limits_{i\in Q_k}v_i\cdot m_{i,k}} \tag{4}$$

(5)审计者对数据持有性证据信息执行批量审计 $BVerify(\{M_{\text{info},k},C_k,sk_{h,k},pk_{t,k},P_k\}_{k\in Uchal})\rightarrow\{0/1\}$:审计者接收到服务器返回的数据持有性证据集合 P,以批量处理的方式验证服务器是否正确存储了用户的文件.

具体过程是,计算标签证据的累乘积 TP,数据证据的累乘积 DP 及文件信息累乘积 $H_{chal,k}$,计算如式(5)～(7)所示.将以上 3 式的计算结果代入式(8)比较等式是否相等,若相等,表明所有文件完好,返回 1;反之,表明至少存在一个失效文件,返回 0.

$$TP=\prod_{k\in Uchal}TP_k \tag{5}$$

$$DP=\prod_{k\in Uchal}DP_k \tag{6}$$

$$H_{chal,k}=\prod_{i\in Q_k}h(sk_{h,k},FID\|i)^{r\cdot v_i} \tag{7}$$

$$DP\cdot\prod_{k\in Uchal}e(H_{chal,k},pk_{t,k})\overset{?}{=}e(TP,g^r) \tag{8}$$

如果批量审计 $BVerify$ 失败,调用识别函数查找失效文件.

(6)审计者查找所有失效文件 $SearchFile(\{M_{\text{info},k},C_k,sk_{h,k},pk_{t,k},P_k\}_{k\in Uchal})\rightarrow\{M_{\text{info},i}\}_{i\in Invalid}$:输入被挑战文件信息,审计者采用查询算法识别出所有失效文件,输出失效文件列表.

当文件审计列表长度为 1(即 $|Uchal|=1$)且批量审计失败时,审计列表中的文件就是一个失效文件.不同失效文件查询算法具有不同的识别过程,在

3.2 节予以详细介绍.简单二分查找 SBS 作为本文的对比算法,描述如下.为了便于论述,作如下定义.

定义 2. 文件证据对 X_k 是由形如$\langle M_k, P_k, U_k\rangle$的三元序偶组成,$M_k$ 为文件信息,P_k 为数据持有性证据,U_k 为用户信息.审计列表中的所有文件证据对构成集合 $X=\{X_k\}_{k\in[1,N]}$.

定义 3. 批量审计函数 $BV(X,V)\rightarrow\{0|1\}$,输入文件证据对集合 X 和标志位 V.如果 $V=$True,对 X 执行批量审计,审计通过返回 True;审计不通过,返回 False;反之,不对 X 执行批量审计,直接返回 False.特别地,当 $V=$True 时,$BV(X,V)$简写为 $BV(X)$.

算法 1. SBS(X,V).

输入:含有 N 个文件证据对的集合 X,标志位 V

输出:失效文件列表

//当 X 中仅含有 1 个文件时,SBS 算法的处理过程

1. IF $N=1$
2. 调用 $BV((M_1,P_1),V)$;
3. IF 审计通过(True)
4. return;
5. ELSE 输出(M_1,P_1), return;
6. ENDIF
7. ENDIF

//当 X 中含有多个文件时,SBS 算法的处理过程

8. IF $N\neq 1$
9. 调用 $BV(X,V)$;
10. IF 审计通过(True)
11. return;

//当批量审计失败后,分裂 X,执行 SBS 的递归调用

12. ELSE 转入步骤 15;
13. ENDIF
14. ENDIF
15. 分裂列表 X 为左右 2 个子节点,左孩子节点 X_L 含有$\lceil n/2\rceil$个文件证据对,右孩子节点 X_R 含有$\lfloor n/2\rfloor$个文件证据对;

//递归查询左、右子树中的失效文件

16. IF SBS(X_L,True)
17. 调用 SBS(X_R,True);
18. ELSE 调用 SBS(X_R,False);
19. ENDIF

例如,文件集合 $M=\{M_1,M_2,\cdots,M_{16}\}$ 中含有若干失效文件$\{M_2,M_7,M_{11},M_{13},M_{14}\}$,在文件证据对集合 $X=\{X_1,X_2,\cdots,X_{16}\}$ 中将对应产生 5 个失效文件证据对$\{X_2,X_7,X_{11},X_{13},X_{14}\}$.批量审计失效后,采用 SBS 方法的查询失效文件的过程如图 2 所示,需要执行 20 次批量审计,依次为$\{X_1,X_2,\cdots,X_{16}\}$、$\{X_1,X_2,\cdots,X_8\}$、$\{X_1,X_2,X_3,X_4\}$、$\{X_1,X_2\}$、$\{X_1\}$、$\{X_3,X_4\}$、$\{X_5,X_6,X_7,X_8\}$、$\{X_5,X_6\}$、$\{X_7\}$、$\{X_8\}$、$\{X_9,X_{10},\cdots,X_{16}\}$、$\{X_9,X_{10},X_{11},X_{12}\}$、$\{X_9,X_{10}\}$、$\{X_{11}\}$、$\{X_{12}\}$、$\{X_{13},X_{14},X_{15},X_{16}\}$、$\{X_{13},X_{14}\}$、$\{X_{13}\}$、$\{X_{14}\}$、$\{X_{15},X_{16}\}$.

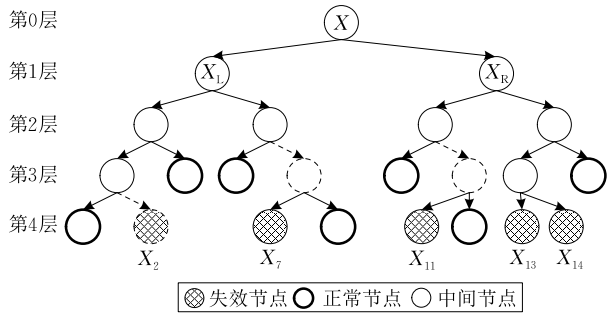

图 2 简单二分查找方法 SBS 的查找过程

3.2 失效文件的快速查询方法

失效文件查询的目标是使用尽可能小的计算开销尽快完成失效文件查询,同时增强系统抵抗"失效文件"攻击的能力.

为了适用不同应用情景,本文提出 3 种失效文件查询方法并分析它们的优缺点.其中,二分快速查找方法(Binary Fast Search,BFS)通用性强,但不太适合"失效文件少但审计文件数量多"的情景;基于幂指测试的二分快速查找方法(Exponents Test Based BFS,ET-BFS)适用于"失效文件少且分散"的情景,但在失效文件聚集处开销较大,不如 BFS.混合型的失效文件快速查询方法(Hybrid BFS,H-BFS)对上述二种算法进行了折衷处理,不仅可以很好利用幂指测试特性有效抵御"失效文件"攻击,并且可以降低"失效文件聚集处"的查询开销.

3.2.1 二分快速查找方法(Binary Fast Search,BFS)

为了充分利用查询过程的中间结果,减少批量审计次数,本文提出二分快速查找方法 BFS,建立批量审计之间的关联性,将无关联审计转换为关联审计.

BFS 方法主要借鉴 Law 等人[17]快速二分查找的思想并利用双线性对映射的性质,将验证过程的"数值比较:$P=Q$"方式转变为"值 1 判断:$A=PQ^{-1}$"方式.具体而言,将批量审计等式(8)等价转换为等式(9).

$$A\overset{?}{=}DP\cdot e(-TP,g^r)\cdot\prod_{k\in U_{chall}}e(H_{chal,k},pk_{t,k}) \quad (9)$$

定义 4. 验证值 A 是式(9)的计算结果,记为 $A=BV(X)$.若 $A=1$,批量审计通过,表明数据完好;若 $A\neq 1$,表明至少存在一个失效文件.特别地,单

个文件证据对 X_k 的验证值 $A_k=BV(X_k)$，如式(10)所示.

$$A_k=DP_k\cdot e(-TP_k,g^r)\cdot e(H_{chal,k},pk_{t,k}) \quad (10)$$

连续多个文件证据对$(i\rightarrow j)$的验证值乘积，表示为 $A_{[i,j]}=\prod_{k=i}^{j}A_k$，根据式(9)文件证据对集合 X 的验证值 $A=A_{[1,N]}=\prod_{k=1}^{N}A_k$.

定理 1.　当且仅当 $A_k=1$，文件证据对 X_k 是有效的；同理，当且仅当 $A=1$，文件证据对集合 X 是有效的.

证明.

充分性. 当 $A_k=1$ 时，表明服务器返回的数据持有性证据 $P_k=\{DP_k,TP_k\}$ 通过了验证函数，表明数据完好，因此文件证据对 X_k 是有效的；

必要性. 如果文件证据对 X_k 是有效的，则批量审计等式(8)恒成立，故 $A_k=1$.　　证毕.

“值 1 判断”方法建立了父节点和左右孩子节点之间的关联性，如图 3 所示. 二分查找树中父节点的验证值等于左右孩子节点验证值的乘积，如式(11)所示. 通过比较父节点和左孩子节点的验证值，可以直接判断右孩子节点的有效性，无需执行批量审计过程.

$$A_{[1,N]}=A_{[1,N/2]}\cdot A_{[N/2+1,N]} \quad (11)$$

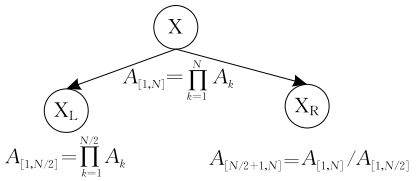

图 3　“值 1 判断”方法

按照基于“值 1 判断”的批量审计函数 $Judge1(X)$ 计算每个节点的验证值，过程如下：

(1) 假定 X 是根节点或者左孩子节点，计算 $Judge1(X)=A_{[1,N]}=BV(X)$；

(2) 若 X 节点失效，即 $Judge1(X)\neq1$，计算 X 的左孩子节点，即 $Judge1(X_L)=A_{[1,N/2]}$；反之，return；

(3) 计算右孩子节点的验证值：如果 $A_{[1,N/2]}=1$，则 $Judge1(X_R)\neq1$；如果 $A_{[1,N/2]}\neq1$，则 $Judge1(X_R)=A_{[N/2+1,N]}=A_{[1,N]}/A_{[1,N/2]}$.

通过以上分析可以看出，基于“值 1 判断”的 BFS 方法，省去了右孩子节点的批量审计过程，从而减少了批量审计次数，算法描述如下.

算法 2.　$BFS(X,V)$.

输入：含有 N 个文件证据对的集合 X，标志位 V

输出：失效文件列表

//当 X 中仅含有 1 个文件时，BFS 算法的处理过程

1. IF $N=1$
2. 　计算 $A=Judge1(X)$；
3. 　IF $A=1$
4. 　　审计通过(True)，return；
5. 　ELSE 输出(M_1,P_1)，return；
6. 　ENDIF
7. ENDIF

//当 X 中含有多个文件时，BFS 算法的处理过程

8. IF $N\neq1$
9. 　计算 $A=Judge1(X)$；
10. 　IF $A=1$
11. 　　审计通过(True)，return；

//当批量审计失败后，分裂 X，执行 BFS 的递归调用

12. 　ELSE 转入步骤 15；
13. 　ENDIF
14. ENDIF
15. 分裂列表 X 为左右 2 个子节点，左孩子节点 X_L 含有 $\lceil n/2\rceil$ 个文件证据对，右孩子节点 X_R 含有 $\lfloor n/2\rfloor$ 个文件证据对；

//递归查询左、右子树中的失效文件

16. IF BFS$(X_L$，True$)$
17. 　　调用 BFS$(X_R$，True$)$；
18. ELSE 调用 BFS$(X_R$，False$)$；
19. ENDIF

例如，文件集合 $M=\{M_1,M_2,\cdots,M_{16}\}$ 中含有若干失效文件 $\{M_2,M_7,M_{11},M_{13},M_{14}\}$，在文件证据对集合 $X=\{X_1,X_2,\cdots,X_{16}\}$ 中将对应产生 5 个失效文件证据对 $\{X_2,X_7,X_{11},X_{13},X_{14}\}$. 批量审计失效后，采用 BFS 方法的查询失效文件的过程如图 4 所示，需要执行 12 次批量审计，虚线节点通过“值 1”判断方法得到. 执行的批量审计依次为 $\{X_1,X_2,\cdots,X_{16}\}$、$\{X_1,X_2,\cdots,X_8\}$、$\{X_1,X_2,X_3,X_4\}$、$\{X_1,X_2\}$、$\{X_1\}$、$\{X_5,X_6\}$、$\{X_7\}$、$\{X_9,X_{10},X_{11},X_{12}\}$、$\{X_9,X_{10}\}$、$\{X_{11}\}$、$\{X_{13},X_{14}\}$、$\{X_{13}\}$.

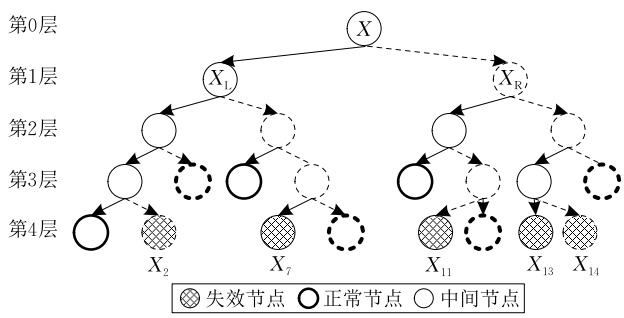

图 4　快速二分查找方法 BFS 的查找过程

与 SBS 方法相比,BFS 方法通过修改右孩子节点的计算方式,有效减少了整个识别过程的批量审计次数. BFS 算法的不足在于失效文件识别过程长度严重依赖于审计文件总数. 即使只有一个失效文件,最少也需要执行$\lceil \log_2 N \rceil$次代价高昂的批量审计.

可以看出,BFS 算法是一种直接但比较"笨拙"的算法,它通用性强,但不太适合"失效文件少但审计文件数量多"的情景. 因此,本文提出了另外一种基于幂指测试的失效文件快速查询方法.

3.2.2 基于幂指测试的二分快速查找方法(ET-BFS)

幂指测试[16]通过计算的方式能够从批量审计文件中快速识别出单个失效文件. 基于幂指测试的二分快速查找方法缩短了识别过程的长度,尽快完成了失效文件的查询,减少了批量审计次数.

定义 5. 幂指验证值 A'_k是验证值 A_k的 k 次方,幂指验证值的计算函数为 $EBV(\cdot)$.

文件证据对 X_k的幂指指验证值 $A'_k=EBV(X_k)$ 计算过程如式(12)所示:

$$\begin{aligned}A'_k&=DP_k^k\cdot e(-TP_k,g^r)^k\cdot e(H_{chal,k},pk_{t,k})^k\\&=DP_k^k\cdot e(-k\cdot TP_k,g^r)\cdot e(k\cdot H_{chal,k},pk_{t,k})\end{aligned}\tag{12}$$

文件证据对集合 X 的幂指验证值 $A'=EBV(X)$ 计算过程如式(13)所示:

$$A'=\prod_{k=1}^{N}A_k^k=e\Big(-\prod_{k=1}^{N}k\cdot TP_k,g^r\Big)\cdot\prod_{k=1}^{N}(DP_k^k\cdot e(k\cdot H_{chal,k},pk_{t,k}))\tag{13}$$

当 X_k有效时,即 $A_k=1$,易得 $A'_k=1$;反之,$A'_k=A_k^k$.

已知文件证据对集合 $X=\{X_k\}_{k\in[1,N]}$,设 I 为失效文件集合,幂指测试的过程如下:

(1) 首先,根据式(9)计算验证值 $A=\prod_{k=1}^{N}A_k$;由定理 1 可知,$A=\prod_{i\in I}A_k$. 特别的,当$|I|=1$时,$A=A_i$.

(2) 其次,计算 $A'=EBV(X)$,如式(13);

由定理 1 得

$$A'=\prod_{k=1}^{N}A_k^k=\prod_{i\in I}A_i^i\tag{14}$$

特别的,当$|I|=1$时,$A'=A_i^i=A^i$.

(3) 累乘 A 并比较 $A'\overset{?}{=}\prod_{l=1}^{L}A$,当 $A'=\prod_{l=1}^{L}A$ 时,累乘次数 L 为失效文件序号,即 X_L无效;如果 L 不存在,表明 X 中至少存在两个失效文件.

节点的幂指验证值按照函数 $ET\text{-}Judge1(X)$计算,过程如下:

(1) 假定 X 是根节点或者左孩子节点,计算 $ET\text{-}Judge1(X)=EBV(X)$;

(2) 如果 X 节点失效,即 $Judge1(X)\neq1$,计算 X 的左孩子节点,即 $ET\text{-}Judge1(X_L)=EBV(X_L)$;反之,return;

(3) 计算右孩子节点的幂指验证值:若 $A'_{[1,N/2]}=1$,则 $ET\text{-}Judge1(X_R)=A'_{[N/2+1,N]}=A'_{[1,N]}\neq1$;反之,$ET\text{-}Judge1(X_R)=A'_{[N/2+1,N]}=A'_{[1,N]}/A'_{[1,N/2]}$.

基于幂指测试的二分查找方法 ET-BFS 提前完成了含有单个失效节点的子树查询,降低了二分查找树的高度,进一步减少了批量审计次数,算法描述如下.

算法 3. ET-BFS(X,V).

输入:含有 N 个文件证据对的集合 X,标志位 V

输出:失效文件列表

//当 X 中仅含有 1 个文件时,ET-BFS 算法的处理过程

1. IF $N=1$
2. 计算 $A=Judge1(X)$;
3. IF $A=1$
4. 审计通过(True), return;
5. ELSE 输出(M_1,P_1), return;
6. ENDIF
7. ENDIF

//当 X 中含有多个文件时,ET-BFS 算法的处理过程

8. IF $N\neq1$
9. 计算 $A=Judge1(X)$;
10. IF $A=1$
11. 审计通过(True),return;
12. ELSE 计算 $A'=ET\text{-}Judge1(X)$;
13. 测试 $A'\overset{?}{=}\prod_{l=1}^{L}A$;
14. IF 存在 L 满足上式
15. 输出(M_L,P_L), return;
16. ELSE IF $N=2$
17. 输出 X;

//批量审计失败后,分裂 X,执行 ET-BFS 的递归调用

18. ELSE 转入步骤 23;
19. ENDIF
20. ENDIF
21. ENDIF
22. ENDIF
23. 分裂列表 X 为左右 2 个子节点,左孩子节点 X_L含有$\lceil n/2\rceil$个文件证据对,右孩子节点 X_R含有$\lfloor n/2\rfloor$个文件证据对;

//递归查询左、右子树中的失效文件

24. IF ET-BFS(X_L,True)
25. 调用 ET-BFS(X_R,True);
26. ELSE 调用 ET-BFS (X_R,False);
27. ENDIF

例如，文件集合 $M=\{M_1,M_2,\cdots,M_{16}\}$ 中含有若干失效文件 $\{M_2,M_7,M_{11},M_{13},M_{14}\}$，在文件证据对集合 $X=\{X_1,X_2,\cdots,X_{16}\}$ 中将对应产生 5 个失效文件证据对 $\{X_2,X_7,X_{11},X_{13},X_{14}\}$. 批量审计失效后，采用 ET-BFS 方法的查询失效文件的过程如图 5 所示，需要执行 5 次批量审计，依次为 $\{X_1,X_2,\cdots,X_{16}\}$、$\{X_1,X_2,\cdots,X_8\}$、$\{X_1,X_2,X_3,X_4\}$、$\{X_9,X_{10},X_{11},X_{12}\}$、$\{X_{13},X_{14}\}$.

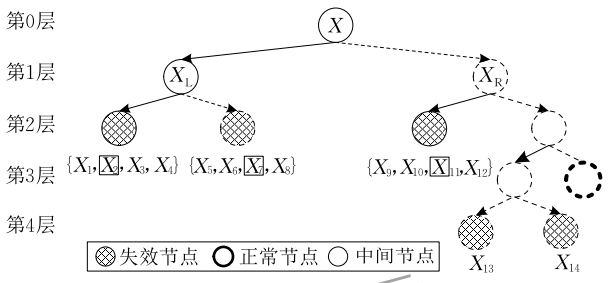

图 5　ET-BFS 的查找过程

3.2.3　混合型二分快速查找方法(H-BFS)

ET-BFS 方法适用于“失效文件少且分散”的情景，但在失效文件聚集处 ET-BFS 方法不如 BFS 方法有效. 为此，本文提出一种混合型失效文件快速查询方法(H-BFS)，该方法不仅可以很好利用幂指测试特性有效抵御“失效文件”攻击，并且可以降低“失效文件聚集处”的查询开销.

H-BFS 方法的核心思想：将二分查找树分为高低两层，以第 α 层分界. 底层部分从第 0 层到第 α 层，采用 ET-BFS 方法查询失效文件；高层部分从第 $\alpha+1$ 层到第 $\lceil\log_2 N\rceil$ 层，采用 BFS 方法查询失效文件.

假设第 α 层中每个节点含有 D 个文件证据对，H-BFS 算法描述如下.

算法 4.　H-BFS(X,V).

输入：含有 N 个文件证据对的集合 X，标志位 V

输出：失效文件列表

//当 X 中仅含有 1 个文件时，H-BFS 算法的处理过程

1. IF $N=1$
2. 　计算 $A=Judge1(X)$；
3. 　IF $A=1$
4. 　　审计通过(True)，return；
5. 　ELSE 输出(M_1,P_1)，return；
6. 　ENDIF
7. ENDIF

//当 X 中含有多个文件时，H-BFS 算法的处理过程

8. IF $N\neq1$
9. 　计算 $A=Judge1(X)$；
10. 　IF $A=1$
11. 　　审计通过(True)，return；
12. 　ELSE IF $|X|\geq D$
13. 　　计算 $A'=ET\text{-}Judge1(X)$；
14. 　　测试 $A'\stackrel{?}{=}\prod_{l=1}^{L}A$；
15. 　　IF 存在 L 满足上式
16. 　　输出(M_L,P_L)，return；
17. 　　ELSE IF $N=2$
18. 　　输出 X；

//批量审计失败后，分裂 X，执行 H-BFS 的递归调用

19. 　　ELSE 转入步骤 25；
20. 　　ENDIF
21. 　ENDIF
22. 　ENDIF
23. 　ENDIF
24. ENDIF
25. 分裂列表 X 为左右 2 个子节点，左孩子节点 X_L 含有 $\lceil n/2\rceil$ 个文件证据对，右孩子节点 X_R 含有 $\lfloor n/2\rfloor$ 个文件证据对；

//递归查询左、右子树中的失效文件

26. IF H-BFS(X_L,True)
27. 　调用 H-BFS(X_R,True)；
28. ELSE 调用 H-BFS (X_R,False)；
29. ENDIF

例如，文件集合 $M=\{M_1,M_2,\cdots,M_{16}\}$ 中含有若干失效文件 $\{M_2,M_7,M_{11},M_{13},M_{14}\}$，在文件证据对集合 $X=\{X_1,X_2,\cdots,X_{16}\}$ 中将对应产生 5 个失效文件证据对 $\{X_2,X_7,X_{11},X_{13},X_{14}\}$. 批量审计失效后，采用 H-BFS 方法的查询失效文件的过程如图 6 所示，需要执行 6 次批量审计，依次为 $\{X_1,X_2,\cdots,X_{16}\}$、$\{X_1,X_2,\cdots,X_8\}$、$\{X_1,X_2,X_3,X_4\}$、$\{X_9,X_{10},X_{11},X_{12}\}$、$\{X_{13},X_{14}\}$、$\{X_{13}\}$.

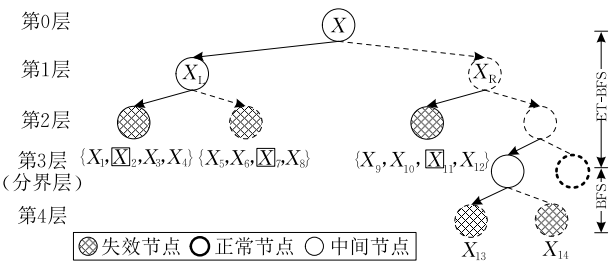

图 6　H-BFS 的查找过程

在本例中，H-BFS 比 ET-BFS 多执行了一次批量审计，但 H-BFS 和 ET-BFS 计算开销相当，因为 H-BFS 的第 3 层、第 4 层无需计算节点的幂指验证值. 可以看出，H-BFS 方法适合抵抗小数量“失效文件”攻击，并可以减少“失效文件聚集处”的查询开销. 在查询大数量“失效文件”时，H-BFS 比 ET-BFS 节省了很多计算幂指验证值的开销，并发挥了 BFS 查询大量失效文件的优势.

4 安全分析

本节首先在挑战信息中添加随机值,保证云存储服务器不可伪造证据,克服了重放攻击和伪造攻击的问题;然后利用双线性对的知识,证明了数据完整性验证的正确性;最后,分析了批量审计的安全性.根据以上分析,可以看出本文方案是满足安全性需求的.

4.1 抵抗伪造攻击和重放攻击

云存储服务器期望使用已知或者过期的信息伪造数据持有性证据 P,但是本文方案中挑战信息中包含伴随随机值$\{v_{i,k},R\}_{i\in Uchal,k}$,即使相同文件挑战相同数据块,随机值也互不相同.由式(3)、(4)计算过程可知,云存储服务器不能依赖历史信息伪造数据持有性证据 P,因此抵挡了恶意云存储服务器的伪造攻击和重放攻击.

4.2 数据完整性验证的正确性

如果审计者和云存储服务器能够按照本方案进行计算,那么云存储服务器生成的证据就能够通过审计者的数据完整性验证.

证明.

由式(8)得

$$\begin{aligned}
\text{左边}&=DP\cdot\prod_{k\in U_{chall}}e(H_{chal,k},pk_{t,k})\\
&=\prod_{k\in U_{chall}}\left[e(u_k,g^{sk_{t,k}\cdot r})^{\sum_{i\in Q_k}v_i\cdot m_{i,k}}\cdot e(H_{chal,k},g^{sk_{t,k}})\right]\\
&=\prod_{k\in U_{chall}}\left[e\Big(\prod_{i\in Q_k}u_k^{v_i\cdot m_{i,k}},g^{sk_{t,k}\cdot r}\Big)\cdot\right.\\
&\quad\left.e\Big(\prod_{i\in Q_k}h(sk_{h,k},FID\|i)^{v_i},g^{sk_{t,k}\cdot r}\Big)\right]\\
&=e\Big(\prod_{k\in U_{chall}}\prod_{i\in Q_k}h(sk_{h,k},FID\|i)\cdot u_k^{m_{i,k}\cdot sk_{t,k}\cdot v_i},g^r\Big)\\
&=e(TP,g^r)=\text{右边}.
\end{aligned}$$

证毕.

4.3 批量审计的安全性分析

批量审计过程可能存在多个失效文件证据对而批量审计通过的现象,但是本文方案每个文件证据对具有直接相关性,不同文件证据对具有不可合并性,这就保证了批量审计的安全性.下面以文件证据对集合 $P_{12}=\{(DP_1,TP_1),(DP_2,TP_2)\}$证明.

证明.

依据式(8)及 3.1 节计算过程,对 P_{12}执行批量审计可以得到

$$\text{左边}=e\Big(\prod_{k\in\{1,2\}}\prod_{i\in Q_k}h(sk_{h,k},FID\|i)\cdot u_k^{m_{i,k}\cdot sk_{t,k}\cdot v_i},g^r\Big),$$

$$\text{右边}=e\Big(\prod_{k\in\{1,2\}}\prod_{i\in Q_k}t_{i,k}^{v_i},g^r\Big).$$

恶意云存储服务器要想失效文件通过批量审计,应使以上二式相等,即有

$$\prod_{i\in Q_k}t_{i,k}^{v_i}=\prod_{i\in Q_k}h(sk_{h,k},FID\|i)\cdot u_k^{m_{i,k}\cdot sk_{t,k}\cdot v_i}.$$

因为云存储服务器无法获知私钥,所以可以保证满足上式等式成立的文件证据信息不可伪造.也就是说,若要是等式成立,当且仅当文件证据对$(DP_1,TP_1),(DP_2,TP_2)$可以分别通过审计. 证毕.

5 性能分析

本文首先通过形式化分析方法,从通信开销和计算开销来衡量了简单二分查找 SBS、基于矩阵查找 MBS 方法和本文方法的性能差异.然后,实现了原型系统,并在真实环境中对上述方法进行了对比分析.

5.1 通信开销

为了验证云端数据的完整性,第三方审计者首先向云存储服务器发送文件集合的挑战信息 $C=\{(\{i,v_i\}_{i\in Q_k},R_k)\}_{k\in[1,N]}$,$N$ 为文件总数.然后,云存储服务器生成并返回数据持有性证据信息 $P=\{(TP_k,DP_k)\}_{k\in[1,N]}$,$k$ 是被审计文件的序号.通信开销分别为 $N\cdot(|Q_k|(|id|+|p|)+|p|)$和 $2N\cdot|p|$(单位为比特),其中 N 是审计文件总数,$|p|$是群 G_1、G_2和 $\mathbb{Z}_p$中元素的大小,$|id|$是数据块标识的大小.

表 1 给出了本文方案和简单二分查找 SBS、基于矩阵查找 MBS 方法的通信开销对比情况.批量审计失败后,本文方案的通信开销与 SBS 方法相同,略高于 MBS 方法的最优情景(仅有一个失效文件损坏时),在绝大多数情景下低于 MBS 方法.

表 1 通信开销比较

方案名称	通信大小/比特
SBS	$N\cdot(\|Q_k\|(\|id\|+\|p\|)+3\|p\|)$
MBS	Best Case: $N\cdot(\|Q_k\|(\|id\|+\|p\|)+\|p\|)+4\sqrt{N}\|p\|$ Worst Case: $N\cdot(\|Q_k\|(\|id\|+\|p\|)+3\|p\|)+4\sqrt{N}\|p\|$
本文方案	$N\cdot(\|Q_k\|(\|id\|+\|p\|)+3\|p\|)$

5.2 计算开销

识别失效文件的计算开销不仅依赖于查询算法,并且受审计负载(审计文件总数、失效文件个数和分布)的影响.

审计者执行一次批量审计(文件总数为 N)的计算开销为

$$Cost_N=\Big[3(N-1)+\sum_{k=1}^{N}(2|Q_k|-1)\Big]\cdot Mul_{G_1}+$$

$$\left(1+\sum_{k=1}^{N}|Q_k|\right)\cdot Exp_{G_1}+$$

$$\left(\sum_{k=1}^{N}|Q_k|\right)\cdot Cost_{\mathrm{Hash}}+(N+1)Pair,$$

其中，Mul_{G_1} 和 Exp_{G_1} 分别表示在群 G_1 上计算一个乘法运算和指数运算的时间，$Cost_{\mathrm{Hash}}$ 是哈希函数的计算时间，$Pair$ 表示计算一个双线性对的时间. 双线性对 $Pair$ 的计算开销远远大于其他运算，为便于讨论，$Cost_N$ 简化为

$$Cost_N \doteq (N+1)\cdot Pair \tag{15}$$

假设文件总数为 N，失效文件数为 w. 在最差情况下，不同查询算法的计算开销（以双线性对计算时间 $Pair$ 为单位）如表 2 所示. 可以看出，MBS 方法计算开销最大，因为 MBS 方法启动开销大（需要对矩阵全行全列执行批量审计），并且在矩阵查询失败后，MBS 需要按照“One by One”方式执行二次查询. BFS 方法采用“值 1 判断”方式节省了一半的批量审计. 虽然 ET-BFS 方法执行每次批量审计的代价是其他方法的 2 倍，但它有效缩短了识别长度，适用于文件数量多且仅有少量文件损坏的情景. H-BFS 方法计算开销略高于 BFS 方法，多出部分为计算底层节点的幂指验证值的开销. H-BFS 方法增强了系统抵抗“失效节点”攻击的能力并且额外的计算开销较少，是 BFS 方法和 ET-BFS 方法之间的一种折衷方案.

表 2　计算开销比较

方法名称	计算开销($Pair$)
SBS	$2(2^{\lceil\log_2 w\rceil}-1+w(\lceil\log_2 N\rceil-\lceil\log_2 w\rceil))(N+1)$
MBS	$(2\sqrt{N}+N)(N+1)$
BFS	$(2^{\lceil\log_2 w\rceil}-1+w(\lceil\log_2 N\rceil-\lceil\log_2 w\rceil))(N+1)$
ET-BFS	$(2^{\lceil\log_2 w\rceil}-1+w(\lceil\log_2 N\rceil-\lceil\log_2(w/2)\rceil))(N+1)$
H-BFS	$(2^{\lceil\log_2 w\rceil}-1+w(\lceil\log_2 N\rceil-\lceil\log_2 w\rceil)+2^{\alpha})(N+1)$

5.3　实验结果分析

为了评估方案的性能，本文利用 PBC 库（pbc-0.5.14）实现了支持失效文件查询的批量审计原型系统 FSBA，并且运行不同失效文件查询方法（SBS、MBS、BFS、ET-BFS，H-BFS）进行对比实验. 原型系统采用 C 语言开发，系统参数为 Ubuntu12.04.3 LTS，Linux 3.8.0-29（x86_64），4x Intel(R) Xeon (R) CPU E5502@1.87GHz，内存 16GB，硬盘 ATA Hitachi HTS54501，150GB.

设置 G_1、G_2、$\mathbb{Z}_p$ 的元素大小为 160 b，即 $|p|=160$，数据块标识大小 $|id|=50$ b，测试文件大小 1GB，数据块大小 4 KB. 假设文件最大损坏率 1%，抽取 460 个数据块进行审计. 为了在审计效率和审计成本之间达到平衡，一般情况下选取 256 个文件执行批量审计（即 $N=256$），这样每次就可以验证大小 2TB+ 的文件集合；失效文件个数 $w=1\sim18$. 为了防止 0～8 个失效文件攻击，在 H-BFS 方法中选取 $\alpha=3$. 测试不同查询算法的通信开销和计算开销，每种情景均测试 100 次以上.

表 3 给出了本文方案和 SBS、MBS 方法的通信开销对比. 本文方法与 SBS 方法的通信开销相同，介于 MBS 方法的最优情况和最差情况之间，但相差不大. 相较于文件大小 1GB，本文的通信开销是可以接受的. 通信开销随文件总数呈线性增长，因此应当选取适中数量的文件进行审计，防止少量文件的损坏带来过大通信开销.

表 3　通信开销实验结果比较

通信开销/KB	文件总数/个				
	16	64	256	1024	4048
SBS	190	758	3034	12135	48540
MBS(Best)	189	757	3025	12098	48385
MBS(Worst)	190	759	3035	12138	48545
本文方案	**190**	**758**	**3034**	**12135**	**48540**
差值	1	1	9	37	155

图 7 显示了在最差情况下，不同查询算法的批量审计次数随失效文件数量的变化（文件总数 256）. 基于矩阵的查询算法 MBS 起步代价大并且要求对失效交叉部分执行二次查询，故 MBS 识别次数最多. 相较于简单二分查找算法 SBS，本文算法 BFS、ET-BFS 和 H-BFS 的批量审计次数减少了近一半，这是由于本文方法利用中间结果计算右孩子节点的验证值，省去了对右孩子节点执行批量审计. ET-BFS 的批量审计次数略高于 BFS，这是因为在最差情景下失效文件成对出现，无法发挥 ET-BFS 快速识别单个失效文件的优势. ET-BFS 多出的审计次数较少，在可接受范围之内. H-BFS 缓解了 ET-BFS 方法问题，在只有少量失效文件出现时，效果最优.

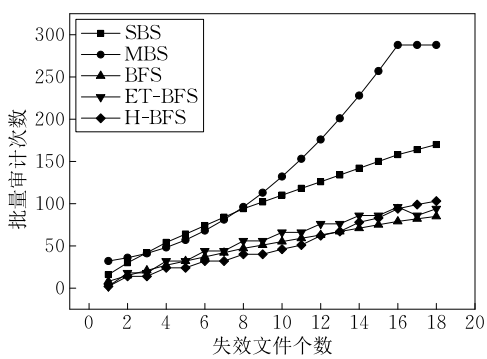

图 7　不同失效文件下批量审计次数比较（最差情况）

图 8 显示了失效文件呈随机分布时，批量审计次数随失效文件数量的变化(审计文件总数 256)．可以看出，本文提出的快速查询算法 BFS、ET-BFS、H-BFS 仍然优于对比算法．在少量文件失效情景，相较于简单二分查找方法，本文方法耗费的批量审计次数减少了 30%．ET-BFS 算法能够及早结束子树查询过程，有效减少了批量审计次数．H-BFS 批量审计次数处于 BFS 和 ET-BFS 之间，以适中的代价完成审计．结合图 7、图 8 可知，H-BFS 方法是对 ET-BFS 的补充，防止 ET-BFS 方法的最差情况出现．

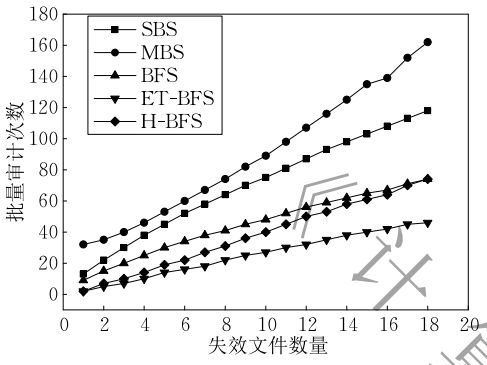

图 8 不同失效文件下批量审计次数比较(随机情况)

设定审计文件数量 256，失效文件数量 1～17，失效文件位置呈随机分布，测试不同查询算法定位所有失效文件的执行时间(ms)，如图 9 所示．基于矩阵的查询算法 MBS 查询时间大于不同类型的 BFS 算法，这是因为 MBS 启动代价太大，需要对所有“行”“列”进行批量审计后再定位．在少量文件损坏时，基于幂指测试方法 ET-BFS 和 H-BFS 查询较快．特别地，仅有一个文件损坏时，ET-BFS 查询时间约为 MBS 和 SBS 算法一半．这是因为，ET-BFS 和 H-BFS 在每次查询过程中使用了幂指测试，使得仅含有 1 个失效文件的查询子路径通过一次计算即可找到该失效文件，而查询时间和批量审计次数联系紧密，在“少量文件”失效时，ET-BFS 批量审计

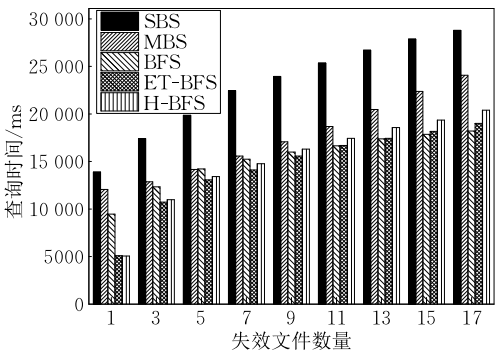

图 9 不同失效文件下查询时间比较(随机情况)

少，查询速度快．并且，随着失效文件增多，BFS 算法执行时间较小，并且增长平缓，优于 SBS 和 MBS 方法．可见，在失效文件较少的情景，本文方案具有优良的查询性能．因此，利用 ET-BFS 算法此种特性，能够有效地抵抗“失效文件攻击”．

为了测试失效文件聚集情景下的查询性能，设置在不同数量的审计文件下 5 个失效文件连续出现，审计文件总数 256，统计不同查询算法的批量审计次数，测试结果如图 10．ET-BFS 批量审计次数接近于 SBS 方法，而 H-BFS 的批量审计次数少于 ET-BFS 方法，降低了“失效文件聚集处”的查询开销．BFS 算法在失效文件聚集处，查询效果最优．这是因为，基于幂指测试的查询算法 ET-BFS 适合于仅含有单失效文件查询路径，对于失效文件聚集处的查询开销较大，不能发挥其优势．混合型查询方法 H-BFS 初始阶段使用 ET-BFS 算法排查失效文件，当发现或者推断存在“失效文件”聚集现象时及时中止 ET-BFS 方法，而采用通用性更强的查询方法 BFS，达到了降低查询开销的目的．在失效文件聚集处，BFS 算法每次查询过程都是有效查询，因此查询效果最好．

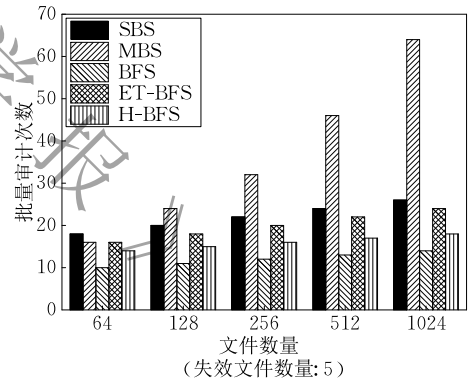

图 10 失效文件聚集处批量审计次数比较

为了验证不同方法抵抗“失效文件”攻击的能力，测试了不同数量失效文件下(失效文件数分别为 1,3,5,7)，批量审计次数随审计文件总数的变化，设置失效文件的位置为随机分布，测试结果如图 11 所示．在少量文件损坏时，本文方法审计次数明显低于 SBS 和 MBS 方法．相较于 BFS 方法，ET-BFS 和 H-BFS 方法进一步减少了识别过程的批量审计次数．特别地，当仅有 1 个失效文件，ET-BFS 和 H-BFS 仅需一次查询即可完成失效文件定位，这种情景中本文方案的查询效率最高．随着失效文件数量增多，批量审计次数随着增多，但是 ET-BFS 算法

增长的最少.这是因为,失效文件数量小且以随机分布的方式散落于各条查询路径,ET-BFS 更容易在较低层次上提前完成失效文件定位,而 SBS、BFS 查询方法需要查找到二分查找树的叶子节点才能完成失效文件定位,因此增加了批量审计次数.在失效文件增多后,混合查询方法 H-BFS 查询代价高于 ET-BFS.这是因为,在低层次无法定位到失效文件时,H-BFS 转变为采用 BFS 算法,因此也会查询到二分查找树的叶子节点,导致查询代价增大.通过以上分析可以看出,本文方案适用于少量失效文件情景下,而"失效文件"攻击正是使用少量失效文件破坏多文件的批量审计.因此,采用本文方法能够执行少量审计即可有效抵御"失效文件"攻击,从而保证了批量审计的可用性和性能.

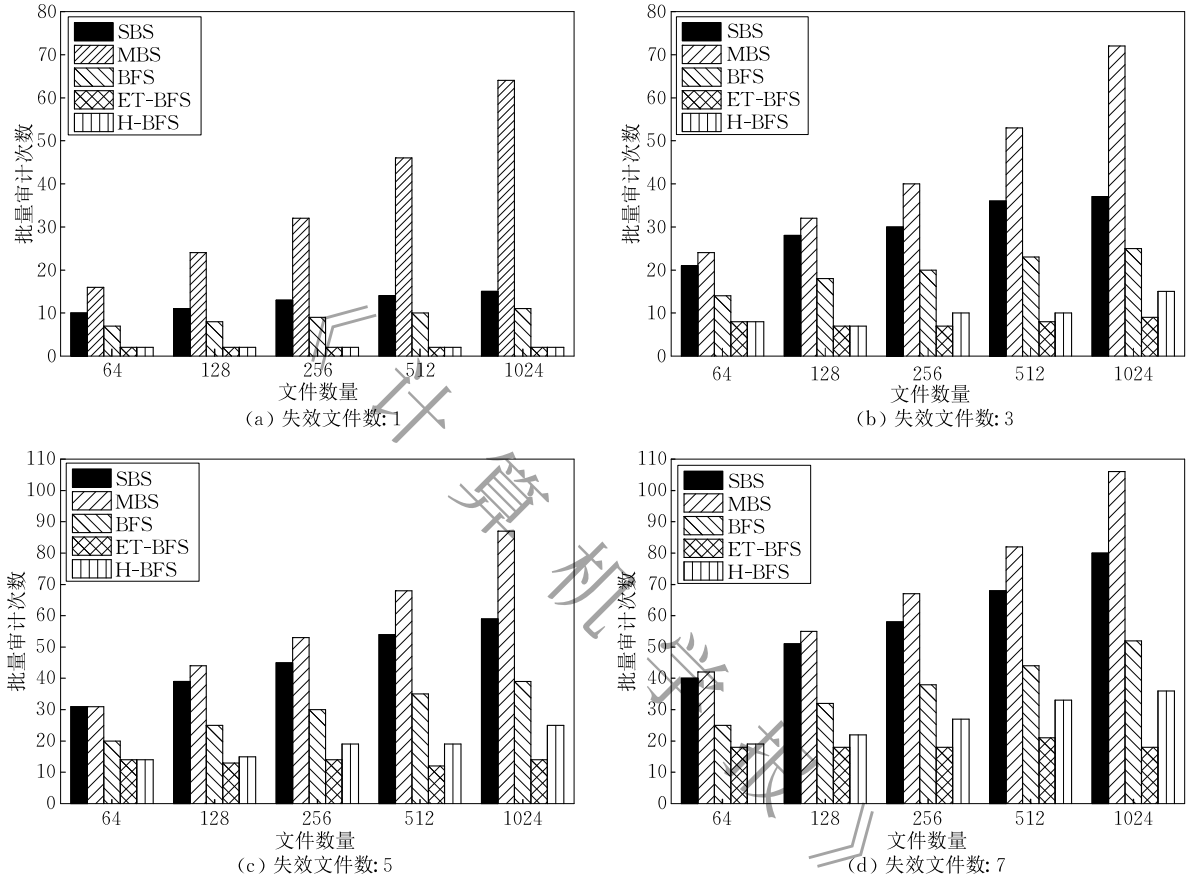

图 11　不同数量文件下批量审计次数比较(随机情况)

6　结束语

本文设计并实现了一种支持失效文件快速查询的批量审计方案 FSBA.该方案不仅支持多用户多文件的批量审计,并且能够在批量审计失败后快速查询失效文件,增强了审计系统的可用性和高效性.特别地,查询方法 BFS 适用于批量审计一般情况,ET-BFS 和 H-BFS 适用于少量文件失效情景,H-BFS 降低了失效文件聚集处的查询开销.ET-BFS 和 H-BFS 增强了系统抵抗"失效文件"攻击的能力.安全分析和实验结果表明本文方案是安全高效的.后续工作将研究差异性文件的批量审计问题并且进一步提高算法的执行效率.

参　考　文　献

[1] Fu Ying-Xun, Luo Sheng-Mei, Shu Ji-Wu. Survey of secure cloud storage system and key technologies. Journal of Computer Research and Development, 2013, 50(1): 136-145 (in Chinese)
(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)

[2] Tan Shuang, Jia Yan, Han Wei-Hong. Research and development of provable data integrity in cloud storage. Chinese Journal of Computers, 2015, 38(1): 164-177(in Chinese)
(谭霜, 贾焰, 韩伟红. 云存储中的数据完整性证明研究及进展. 计算机学报, 2015, 38(1): 164-177)

[3] Ateniese G, Burns R, Curtmola R, et al. Provable data possession at untrusted stores//Proceedings of the 14th ACM

Conference on Computer and Communications Security. New York, USA, 2007: 598-609

[4] Erway C, Küpçü A, Papamanthou C, et al. Dynamic provable data possession//Proceedings of the 16th ACM Conference on Computer and Communications Security. New York, USA, 2009: 213-222

[5] Wang Cong, Wang Qian, Ren Kui, et al. Privacy-preserving public auditing for data storage security in cloud computing//Proceedings of the IEEE International Conference on Computer Communications. San Diego, USA, 2010: 525-533

[6] Zhu Yan, Ahn G J, Hu Hong-Xin, et al. Dynamic audit services for outsourced storages in clouds. IEEE Transactions on Services Computing, 2013, 6(2): 227-238

[7] Wang Hua-Qun, Zhang Yu-Qing. On the knowledge soundness of a cooperative provable data possession scheme in multicloud storage. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(1): 264-267

[8] Ateniese G, Burns R, Curtmola R, et al. Remote data checking using provable data possession. ACM Transactions on Information and System Security, 2011, 14(1): 1-34

[9] Wang Cong, Chow S S M, Wang Qian, et al. Privacy-preserving public auditing for secure cloud storage. IEEE Transactions on Computers, 2013, 62(2): 362-375

[10] Zhu Yan, Hu Hong-Xin, Ahn G J, et al. Cooperative provable data possession for integrity verification in multicloud storage. IEEE Transactions on Parallel and Distributed Systems, 2012, 23(12): 2231-2244

[11] Yang Kan, Jia Xiao-Hua. An efficient and secure dynamic auditing protocol for data storage in cloud computing. IEEE Transactions on Parallel and Distributed Systems, 2013, 24(9): 1717-1726

[12] Wang Bo-Yang, Li Bao-Chun, Li Hui. Oruta: Privacy-preserving public auditing for shared data in the cloud. IEEE Transactions on Cloud Computing, 2014, 2(1): 43-56

[13] Fu Yan-Yan, Zhang Min, Chen Kai-Qu, et al. Proofs of data possession of multiple copies. Journal of Computer Research and Development, 2014, 51(7): 1410-1416(in Chinese)
(付艳艳, 张敏, 陈开渠等. 面向云存储的多副本文件完整性验证方案. 计算机研究与发展, 2014, 51(7): 1410-1416)

[14] Hwang M S, Lee C C, Sun T H. Data error locations reported by public auditing in cloud storage service. Automated Software Engineering, 2014, 21(3): 373-390

[15] Ren Yan-Li, Wang Shuo-Zhong, Zhang Xin-Peng, et al. An efficient batch verifying scheme for detecting illegal signatures. International Journal of Network Security, 2015, 17(4): 463-470

[16] Shin S, Kim S, Kwon T. Identification of corrupted cloud Storage in batch auditing for multi-cloud environments//Proceedings of the Information and Communication Technology-Eurasia Conference. Daejeon, Korea, 2015: 221-225

[17] Law L, Matt B J. Finding invalid signatures in pairing-based batches//Proceedings of the 11th IMA International Conference on Cryptography and Coding. Cirencester, UK, 2007: 34-53

[18] Matt B J. Identification of multiple invalid signatures in pairing-based batched signatures//Proceedings of the International Conference on Practice and Theory in Public Key Cryptography. Irvine, USA, 2009: 337-356

[19] Bernstein D J, Doumen J, Lange T, et al. Faster batch forgery identification//Proceedings of the 13th International Conference on Cryptology. Kolkata, India, 2012: 454-473

WANG Hui-Feng, born in 1986, Ph. D. candidate. His research interests include massive data storage and cloud storage security.

LI Zhan-Huai, born in 1961, professor, Ph.D. supervisor. His main research interests include database theory and technology and massive data storage.

ZHANG Xiao, born in 1978, Ph. D., associate professor. His main research interests include massive data storage and green storage.

SUN Jian, born in 1982, Ph. D. candidate. His main research interests include massive data storage and green storage.

ZHAO Xiao-Nan, born in 1979, Ph. D., lecturer. Her main research interests include massive data storage and hierarchical storage.

Background

With the rapid development of cloud computing, cloud storage service (CSS) has become an important new type data storage model. CSS provides an online data storage service with the form of outsource, which has the advantages of pay-as-you-go, flexible scalability, easy access and so on. CSS greatly reduces the data storage costs and effectively improve the efficiency of resource utilization rate. However, data integrity in the cloud storage is vulnerable to be affected by cloud security incidents. Users can not confirm whether the data in the cloud is intact or not, so that it becomes a major

reason for users to use less or no CSS. Moreover, the "hidden" damage data can weaken the data recovery ability which affects data storage security. It requires an independent auditing service to check the data integrity in the cloud, which can eliminate the user concerns and enhance the cloud storage security. Recently, several data auditing protocols have been proposed, which focus on dynamic data auditing, privacy-preserving auditing, batch auditing and robust auditing.

In particular, batch auditing is the key technology to efficiently verify data integrity in the cloud storage service. However, the batch audit schemes are vulnerable to "invalid files" attacks. Searching invalid files bring heavy cost and the search speed is slow, which seriously affect the availability and efficiency of batch audit. To solve this problem, we propose a batch audit scheme with fast searching invalid files (FSBA) in cloud storage service. Through establishing the relationship of batch audit process, it can effectively reduce the number of batch verification in the whole search process when batch auditing fails. It also designs two other methods to improve the search performance of specific application scenarios, which has small little invalid files in a batch auditing.

Our team has been working on the research of data integrity checking and has achieved certain results, such as the delayed update method for improving the update efficiency; the self-adaptive audit method to enhance the flexibility of auditing process; the multi-proxies method to improve the availability and relieve the performance bottleneck. This paper is based on previous works, which designs a batch auditing system supporting fast searching invalid files to enhance the availability.

This research is supported by the National High Technology Research and Development Program (863 Program) of China (No. 2013AA01A215), the National Natural Science Foundation of China (Nos. 61472323, 61502392), the Specialized Fund for the Basic Research Operating Expenses Program of Central College (No. 3102015JSJ0009) and the Huawei Innovation Research Program (No. YB2014040023).