文章编号：1000-8608(2018)02-0187-07

基于动态布隆过滤器的云存储数据持有性验证方法

谢丽霞*， 胡立杰

（中国民航大学 计算机科学与技术学院，天津 300300）

摘要： 针对现有云存储数据持有性验证方法验证效率低的问题，提出一种基于动态布隆过滤器的云存储数据持有性验证方法. 首先使用同态哈希函数对云存储数据进行处理，简化客户端计算量；然后第三方验证平台使用数据块标签构造动态布隆过滤器，支持云存储数据的全动态操作；最后通过随机验证路径生成持有证据，增强云存储数据持有性验证的安全性. 实验结果表明该方法可有效减少证明计算开销，提高了验证安全性和验证效率.

关键词： 云存储；持有性验证；动态布隆过滤器；动态操作；验证路径

中图分类号： TP393 **文献标识码：** A **doi：** 10.7511/dllgxb201802012

0 引 言

大数据时代，作为云计算的基础，云存储发展迅猛. 但是，日益严重的安全问题已经制约了云存储健康发展[1-2]. 其中，云存储数据的完整性是不可忽略的一个重要问题. 例如：外部攻击或云服务提供商内部人员恶意操作导致数据完整性遭到破坏；云服务提供商为维护自身名誉，隐瞒数据丢失或损坏. 因此，研究一种可公开验证的云存储数据持有性验证方法是很有必要的[3].

为保护云存储数据的完整性，国内外研究者们提出了多种数据持有性验证方案. Ateniese 等[4]提出一种数据持有性证明（provable data possession，PDP）方案，该方案虽然可降低通信开销，但是仅适用于静态数据的验证，并未考虑动态数据的更新问题. Wang 等[5]提出一种基于同态加密短消息签名的动态数据完整性验证方法，该方法支持公开验证和动态数据操作，但当校验元数据规模变大后插入操作的开销将十分巨大. Li 等[6]提出一种基于双线性组的完整性验证方法，基于 Hellman 计算困难问题[7]构造双线性映射用于校验元数据计算，降低客户端执行验证协议初始化阶段的成本，但是由于验证过程使用的结构复杂导致验证效率降低. Hussien 等[8]使用双块传输和加密散列函数的方式验证云存储数据的完整性，可减少客户端计算量，但是却导致辅助存储空间增大，隐私泄露的风险提高. Zhang 等[9]使用 rb2-3 树作为验证工具实现数据公开验证和动态数据完整性验证，但该方法仍存在计算复杂、验证路径过长的问题. 李勇等[10]和 Zhang 等[11]使用树状验证路径来确定数据块位置的正确性，减少各实体间的计算负担，简化动态更新过程，但是在云端计算持有证据时需要更多的辅助信息，导致云端与验证端的通信开销增加.

针对上述方法存在持有性证明计算复杂且验证过程需要大量辅助存储空间的不足，本文提出一种基于动态布隆过滤器（dynamic Bloom filter，DBF）的云存储数据持有性验证方法. 该方法使用同态哈希函数处理验证数据并生成用于验证的校验元，基于数据块标签构造动态布隆过滤器且支持动态操作，以实现对云存储数据持有性的高效验证.

1 动态布隆过滤器

1.1 BTBF 结构

本文提出的 DBF 为 B-Tree 状的布隆过滤

收稿日期： 2017-10-09； **修回日期：** 2018-01-09.
基金项目： 国家科技重大专项资助项目（2012ZX03002002）；国家自然科学基金资助项目（60776807，61179045）；中国民航科技基金资助项目（MHRD201009，MHRD201205）.
作者简介： 谢丽霞*（1974-），女，副教授，E-mail：lxxie@126.com；胡立杰（1991-），男，硕士生，E-mail：ljhu1987@hotmail.com.

器，即 B-Tree 布隆过滤器(B-Tree Bloom filter，BTBF). BTBF 不会因数据大规模增长而线性退化且有较高的查找效率，与现有云存储数据持有性验证方法相比，同样树高的 BTBF 可存储更多校验元.

BTBF 每个节点包含 i 个 BF，除根节点外，BTBF 每个节点可存储多个关键字，一个 4 阶 BTBF 如图 1 所示.

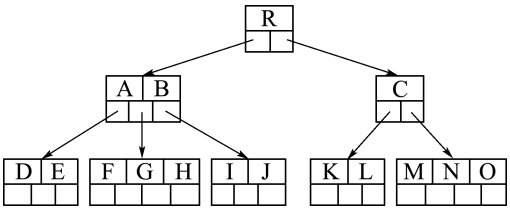

图 1 BTBF 结构

Fig. 1 BTBF structure

本文提出的 BTBF 需满足以下 7 个条件：

(1) BTBF 中节点的关键字由数据块标签、对应的验证签名构成，其中验证签名包括数据块签名和验证计数位；

(2) BTBF 中节点按数据块标签从大到小排列；

(3) 任意非叶子节点最多只有 M 个子节点，且 $M>2$，其中 M 为 BTBF 容度；

(4) 根节点的子节点数为$[2,M]$；

(5) 根节点外的非叶子节点数为$[M/2,M]$；

(6) 每个节点存放至少$\lfloor M/2-1 \rfloor$个和至多 $M-1$ 个布隆过滤器；

(7) 非叶子节点的关键字数量比对应子节点少 1.

1.2 BTBF 的错误率

本文提出的 BTBF 在云存储数据持有性证明中错误率为固定值. 传统布隆过滤器存在错误率和饱和度的问题[12]，由于难以预知集合中的元素数量，随着插入元素的增多饱和度会增加，错误率也将随之提高并最终导致布隆过滤器不可用. BTBF 错误率 P 可由特征值数量 n、数位组长度 m 和选择哈希函数的数量 k 决定，P 与各参数之间的关系为

$$P=(1-\mathrm{e}^{-k\frac{m}{n}})^k \tag{1}$$

云存储数据持有性证明中因为数据分块大小固定，因此通过选择适当的 m、n 和 k，可保证 BTBF 错误率变为一个可控的固定值，避免错误率升高.

2 云存储数据持有性验证方法

2.1 验证模型

传统的完整性验证模型不支持公开验证且需要在客户端存储大量数据[13]，因此本文在传统云存储验证模型的基础上引入第三方验证平台，云存储数据完整性验证模型结构如图 2 所示.

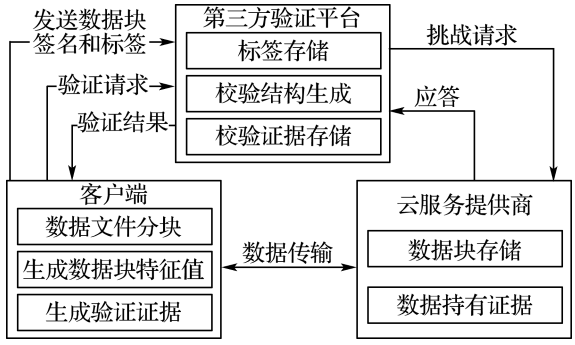

图 2 云存储数据完整性验证模型

Fig. 2 Cloud storage data integrity verification model

第三方数据完整性验证模型包含客户端(Client Server，CS)、云服务提供商(Cloud Server Provider，CSP)和第三方验证平台(Third Party Verification，TPV)，CS 将数据加密上传并生成校验证据，CSP 存储数据和生成数据持有证据应答，TPV 存储校验证据和进行数据完整性验证.

2.2 方法具体实现

基于 DBF 的云存储数据持有性证明方法包括 3 个阶段：初始化阶段、挑战阶段和应答阶段. 3 个阶段的具体设计如下：

(1)初始化阶段

首先，CS 对存储数据进行预处理，然后，将数据和校验证据分别上传给 CSP 和 TPV，过程设计如下：

(a)CS 将数据 $\boldsymbol{F}$ 分割为大小固定的数据块 $m_1,m_2,\cdots,m_v$，对数据块 $m_i(i=1,2,\cdots,v)$ 进行预处理，生成对应数据块的特征值向量 $\boldsymbol{A}_i=(a_{i1}\quad a_{i2}\quad\cdots\quad a_{iv})(i=1,2,\cdots,v)$，通过 k 个同态哈希

函数生成数据块验证签名 bf_i（bf_i 为 BF 生成的数位组，$i=1,2,\cdots,v$），然后将数据块 $m_1,m_2,\cdots,m_v$ 和请求 $CREAT=\{(num_1:bf_1),(num_2:bf_2),\cdots,(num_v:bf_v)\}$ 分别上传到 CSP 和 TPV，上传结束对数据进行一次完整性校验，校验成功后 CS 将删除本地存放的数据.

(b)CSP 存储 CS 上传的数据块 $m_1,m_2,\cdots,m_v$，TPV 接受 $CREAT$ 请求后存储数据块验证签名集 $bf_1,bf_2,\cdots,bf_v$，初始化验证计数位并生成 BTBF，要求 BTBF 容度 $M\geqslant 4$. BTBF 的关键字由数据块验证签名 bf_i 和数据块标签 num_i 组成.

(2)挑战阶段

初始化完成后，CS 可发起挑战请求 CHL，TPV 生成验证请求 TPV_CHL. 过程设计如下：

CS 发起挑战请求 CHL，TPV 接受请求后在 BTBF 验证树中随机选择一条或多条验证路径 $\boldsymbol{S}=(num_1\quad num_2\quad\cdots\quad num_s)$，其中 $num_i(i=1,2,\cdots,s)$ 为数据块标签，数据验证率 $c\geqslant 80\%$，即验证块数量需大于数据块总数的 80%. TPV 将生成的随机路径 $\boldsymbol{S}$ 封装为验证请求 $TPV\_CHL=\{\boldsymbol{S}_1,\boldsymbol{S}_2,\cdots,\boldsymbol{S}_s\}$ 发送给 CSP，要求 CSP 提供验证请求中包含数据块的持有证据.

(3)应答阶段

CSP 在接受验证请求后，根据 TPV_CHL 生成对应数据块持有证据交由 TPV 进行持有性验证，过程设计如下：

CSP 接受 TPV_CHL 请求后，根据请求中包含的数据块标签 num_i 查找对应的数据块 m_i，生成校验元向量 $\boldsymbol{A}_i=(a_{i1}\quad a_{i2}\quad\cdots\quad a_{iv})(i=1,2\cdots,v)$，通过 k 个同态哈希函数生成长度为 s 的数据持有证据 cf_i，生成应答 $\boldsymbol{R}=(cf_1\quad cf_2\quad\cdots\quad cf_s)$ 并反馈给 TPV，TPV 计算校验结果 α：

$$\alpha=\sum_{i=1}^{s}bf_i\oplus cf_i \tag{2}$$

其中 s 为随机验证路径长度. 若 $\alpha=0$，则云存储数据完整性验证成功，TPV 向 CS 返回结果“TRUE”，否则验证失败，返回结果“FALSE”，BTBF 节点参与验证后需对该节点的校验计数位 bc 进行自增操作.

2.3　数据动态操作

基于 DBF 的数据持有性验证方法可支持包括更新、插入和删除的数据操作.

2.3.1　*数据更新算法*　当用户更新数据块 m_i（$i=1,2,\cdots,v$）时，BTBF 云存储数据持有性验证方法的处理过程设计如下：

(a)CS 计算更新数据块 m_i 的标签值 num_i 和签名 bf_i，分别向云存储服务器和 TPV 发送更新请求 $Update\_C=\{num_i:m_i\}$ 和 $Update\_T=\{num_i:bf_i\}$.

(b)云存储服务器接收更新请求 $Update\_C=\{num_i:m_i\}$ 后，根据标签值 num_i 寻找存储在云端的数据块，用 m_i 替换原有数据块 m_i'.

(c)TPV 接收更新请求 $Update\_T=\{num_i:bf_i\}$ 后，根据 num_i 在 BTBF 中查找对应的校验数位组并替换为 bf_i，并将节点验证计数位 bc_i 初始化为 0.

(d)当云存储服务器和 TPV 进行更新操作后，CS 对云存储服务器进行一次数据完整性校验，校验成功后 CS 删除本地存储的 m_i.

2.3.2　*数据插入算法*　当用户插入新数据块 m_i（$i=1,2,\cdots,v$）时，BTBF 云存储数据持有性验证方法的处理过程设计如下：

(a)CS 计算插入数据块 m_i 的标签值 num_i 以及签名 bf_i，分别向云存储服务器和 TPV 发送插入请求 $Insert\_C=\{num_i:m_i\}$ 和 $Insert\_T=\{num_i:bf_i\}$.

(b)CSP 接收请求 $Insert\_C=\{num_i:m_i\}$ 后，存储接收的数据块 m_i.

(c)TPV 接收插入请求 $Insert\_T=\{num_i:bf_i\}$ 后，首先在 BTBF 中根据 num_i 查找数据块签名 bf_i 的插入位置，然后将 bf_i 插入到节点中. 插入完成后需对插入的节点进行判断，若该节点存放的数据块签名数量大于 M，则需分裂该节点并对 BTBF 进行调整，保证在完成插入操作后仍能满足 BTBF 的 7 个条件.

(d)当 CSP 和 TPV 全部进行插入操作后，CS 对云存储服务器进行一次数据完整性校验，校验成功后 CS 删除本地存储的 m_i.

2.3.3　*数据删除算法*　当用户删除数据块 m_i（$i=1,2,\cdots,v$）时，BTBF 云存储数据持有性验证方法的处理过程设计如下：

(a)CS查找删除数据的标签值 num_i,分别向云存储服务器和TPV发送删除请求 $Delete_C=\{num_i\}$和 $Delete\_T=\{num_i\}$.

(b)云存储服务器接收 $Delete_C=\{num_i\}$后,根据数据块标签 num_i 查找数据块并删除.

(c)TPV接收 $Delete_T=\{num_i\}$后,在BTBF中查找数据块标签为 num_i 的节点并删除该节点中对应的关键字.节点在数据块删除后,需要判断该节点关键字数量是否满足每个节点至少存放$\lfloor M/2-1 \rfloor$和至多 $M-1$ 的条件.如果该节点不满足,BTBF树需要对该节点和邻近节点进行合并,使得树仍满足BTBF条件.

(d)当CSP和TPV分别进行删除操作后,CS对云存储服务器进行一次数据完整性校验,校验成功后CS删除操作完成.

本文提出的基于DBF的云存储数据持有性验证方法的动态操作与传统方法相比具有以下特点:

(1)更新操作不只在叶子节点处进行,BTBF树的内部均可进行更新操作,优化存储空间.

(2)删除操作只针对删除节点,不需对其他节点进行更新操作,减少删除造成的冗余计算.

(3)与常用的默克尔哈希树相比,BTBF的插入操作优化验证树的高度,避免大量执行插入操作后验证树退化为线性结构,提高数据完整性验证效率.

3 验证的安全性分析

对基于DBF的云存储数据持有性验证方法分别从验证正确性、数据保密性、持有证明不可伪造性和抗重放攻击等几方面进行分析.

3.1 正确性分析

BTBF在云存储数据持有性验证中存在由假阳性问题造成的错误率,因此本文采用多个数位组构成验证路径的方式降低错误率,使得验证的正确率达到100%,布隆过滤器的错误率[14]如表1所示.

设BTBF高为 h,容度为 M,随机验证路径 $\boldsymbol{S}$ 的长度为 s,单个BTBF节点验证错误率为 f.则数据验证率 $c=sh/(M^h-1)$,单条验证路径错误率 $P_s=f^{sh}$.验证过程中,当 $sh\geqslant 3$ 时,若 $f\leqslant 0.01$,则 $P_s\leqslant 1\times 10^{-6}$.

表1 布隆过滤器错误率

Tab.1 Bloom filter's error rate

m/n	P				
	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
8	0.118 0	0.048 9	0.030 6	0.024 0	0.021 7
9	0.105 0	0.039 7	0.022 8	0.016 6	0.014 1
10	0.095 2	0.032 9	0.017 4	0.011 8	0.009 4

在本文方法中,f 设为$\leqslant 0.01$ 的固定值,同时设定 $c\geqslant 80\%$,则 $sh\geqslant 0.8(M^h-1)$.为提高验证效率设定 $M\geqslant 4$,此时得到 $sh\geqslant 3$.因此本文提出的方法错误率 $P_s\leqslant 1\times 10^{-6}$,趋近于0.

可见,本文方法使用BTBF验证正确率可达到100%.

3.2 保密性分析

第三方平台通过两种方式得到用户有效数据:方式一为第三方平台通过CS上传的校验证据,方式二为CSP提供的持有证据.

首先,CS将数据文件 $\boldsymbol{F}$ 分为固定大小的数据块 $m_1,m_2,\cdots,m_v$,数据块 m_i 生成校验元向量 $\boldsymbol{A}_i=(a_{i1}\quad a_{i2}\quad\cdots\quad a_{iv})(i=1,2,\cdots,v)$,通过 k 个哈希函数生成长度为 m 的数据块签名 bf_i.由于 $\boldsymbol{A}_i$ 为 m_i 的特征值并只含有部分原始数据,经过 k 个哈希函数生成数位组后,为仅包含0和1的序列.由于哈希过程不可逆,攻击方不能通过 bf_i 逆向得到原始数据,因此可保证有效数据不被TPV获得.

其次,CSP提供的持有性证明 cf_i 同样是经过哈希函数处理后的数位组,这是个不可逆的过程,因此攻击者无法通过持有性证明获得有效数据.

所以,本文方法可防止数据隐私泄露,保证TPV不能得到客户有效数据.

3.3 抗伪造攻击和抗重放攻击分析

首先,假设攻击方伪造数据持有证据($\boldsymbol{R}_p=(cf'_1\quad cf'_2\quad\cdots\quad cf'_s)$),其中,$cf'_i(i=1,2,\cdots,s,s$ 为验证路径 $\boldsymbol{S}$ 长度)为数据标签.其中对于BTBF单个节点关键字伪造成功的成功率 $P_1=1/2^m$,那么单条随机验证路径 $\boldsymbol{S}$ 的持有证据伪造的成功率

为 P_s：

$$P_s = P_1^s = \left(\frac{1}{2^m}\right)^s \tag{3}$$

当攻击方对多条随机路径发起联合攻击时，多条路径的持有证据需同时通过 TPV 的持有验证，因此对于 w 条随机路径联合攻击成功概率 P_c 可通过 P_s 得到：

$$P_c = P_s^w = \left(\frac{1}{2^m}\right)^{ws} \tag{4}$$

w 条随机路径联合攻击的成功概率 P_c 会随着 s 和 w 的增大而逐渐减小，由于 $m \geqslant 6$，则 P_c 远小于 0.015 6，可认为攻击方对多条路径联合攻击的成功概率为 0. 因此，本文提出的方法，当验证的数据规模越大时，抵抗伪造攻击的能力越强.

本文的云存储数据完整性验证方法为动态验证方法，所以 TPV 存储的校验数据会随时间变化而变化. 当攻击方在 BTBF 树更新后进行攻击时，由于 BTBF 验证树会因为更新操作发生改变，重放已有的持有证据并不能通过 TPV 的完整性验证；当攻击方在 BTBF 树未更新时进行重放攻击，由于 BTBF 关键字中包括验证计数位，在验证计数位未完全消耗的情况下，每次验证时验证计数位不同，因此重放已有的持有证据，并不能通过 TPV 的完整性验证. 当在 BTBF 节点计数位未消耗完毕时，本文提出的方法可抵抗重放攻击.

4　实验与结果

本文主要针对基于 DBF 的云存储数据持有性验证方法进行实验，实验重点为持有性证明计算开销、辅助存储空间和 BTBF 性能. 在 Linux 下使用 Python 语言实现本文模型的核心功能，其中动态操作和 BF 计算使用 Python 的 Btrees 模块和 PybloomFiltermmap 模块实现，实验数据集为 47 000 个随机生成的数据文件，数据集大小为 2 048 MB.

4.1　计算开销

为验证数据持有性证明计算更简单，分析并计算 BTBF 方法、MHT 方法[5]、rb2-3 方法[9]以及 UpdateTrees 方法[11]的各项时间复杂度，记录如表 2(其中 n 为数据块数量，K 为常数，D 为动态操作的数量).

由表 2 所见，BTBF 方法与 rb2-3 方法的 CSP 时间复杂度最小，都可以达到常量级；但是本文提出的 BTBF 方法在树操作时间复杂度上明显优于其他几种方法.

然后对 BTBF 和 rb2-3 两种方法 CSP 的计算时间进行对比，设置数据分块大小为 5 MB，BTBF 数位组长度为 20 000，单个 BTBF 节点错误率为 0.1%. 在实验中，统计并记录 100、200、300、400 和 500 块数据块下，两种方法 CSP 计算持有证据的时间，结果取 10 次平均值，实验结果如图 3 所示.

表 2　验证方法时间复杂度对比

Tab. 2　Verification method's time complexity comparison

验证方法	CSP 时间复杂度	TPV 时间复杂度	树操作时间复杂度
BTBF	$O(K)$	$O(\log n)$	$O(K)$
MHT	$O(\log n)$	$O(\log n)$	—
rb2-3	$O(K)$	$O(\log n)$	$O(\log n)$
UpdateTrees	$O(\log n)$	$O(\log D)$	$O(\log D)$

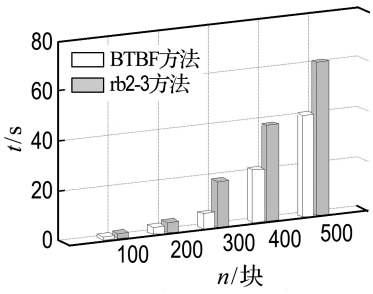

图 3　不同数据量的持有证据计算时间

Fig. 3　The calculation time of different data volume of provable data possession verification

由图 3 可见：本文提出的 BTBF 方法持有性证明的计算时间明显少于 rb2-3 方法，并且随着处理规模的增大，时间差异逐渐增大. 本文提出的 BTBF 方法计算更加简单，计算时间开销更小.

4.2　辅助存储空间

对比 BTBF 方法和 rb2-3 方法的辅助存储空间，设置单个 BTBF 节点错误率为 0.1%，数位组长度为 20 000，数据分块大小为 5 MB；分别统计 BTBF 方法和 rb2-3 方法在 128、256、512、1 024 和 2 048 MB 数据量下的辅助存储，结果如表 3 所

示.

从表3可见:(1)相同数据规模下BTBF使用的辅助存储空间远小于rb2-3方法;(2)BTBF的辅助存储空间与所验证的数据规模大小呈线性相关,辅助存储空间大小可控.本文提出的BTBF方法具有更优的辅助存储,可提高云存储数据的完整性验证效率.

表3 辅助存储开销对比

Tab.3 Auxiliary storage's overhead comparison

数据大小/MB	rb2-3辅助存储空间/MB	BTBF辅助存储空间/MB
128	2.250	0.095
256	5.200	0.191
512	9.000	0.382
1 024	19.000	0.764
2 048	36.000	1.500

4.3 BTBF性能

为验证BTBF的性能,分别对BTBF生成和查询时间进行实验,设置数据分块大小为5 MB,BTBF数位组长度为20 000,单个BTBF节点错误率为0.1%.在实验中,获取并统计BTBF生成和查询时间,结果取平均值,实验结果如图4所示.

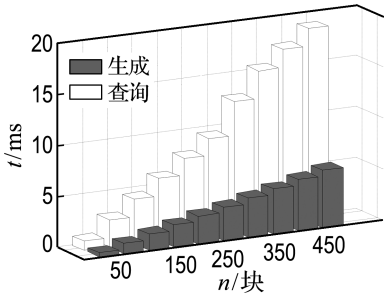

图4 BTBF生成和查询时间

Fig.4 BTBF generation and query time

从图4可见:(1)BTBF验证树的生成和查询时间降低到20 ms以下,计算时间有效降低;(2)BTBF的生成时间增长速率较缓慢,不会因需校验的数据量变大而导致生成时间快速增长.BTBF避免了因为数据规模增大而线性退化最终导致验证效率降低的问题.

5 结 语

本文提出一种基于DBF的云存储数据持有性验证方法.该方法通过使用动态布隆过滤器处理特征值向量生成数据块标签,减少客户端计算量;使用随机验证路径对云存储数据进行持有性验证;将TPV存储空间变成与校验元数据规模相关的值,并且支持数据全动态操作.实验表明该方法提高了云存储数据持有性验证效率.

参考文献:

[1] 冯登国,张 敏,张 妍,等. 云计算安全研究[J]. 软件学报, 2011, **22**(1):71-83.
FENG Dengguo, ZHANG Min, ZHANG Yan, *et al*. Study on cloud computing security [J]. **Journal of Software**, 2011, **22**(1): 71-83. (in Chinese)

[2] ALI M, KHAN S U, VASILAKOS A V. Security in cloud computing: Opportunities and challenges [J]. **Information Sciences**, 2015, **305**(1):357-383.

[3] 谭 霜,贾 焰,韩伟红. 云存储中的数据完整性证明研究及进展[J]. 计算机学报, 2015, **38**(1):164-177.
TAN Shuang, JIA Yan, HAN Weihong. Research and development of provable data integrity in cloud storage [J]. **Chinese Journal of Computers**, 2015, **38**(1):164-177. (in Chinese)

[4] ATENIESE G, BURNS R, CURTMOLA R, *et al*. Provable data possession at untrusted stores [C] // **CCS'07: Proceedings of the 14th ACM Conference on Computer and Communications Security**. NY: ACM Press, 2007:598-609.

[5] WANG Qian, WANG Cong, LI Jin, *et al*. Enabling public verifiability and data dynamics for storage security in cloud computing [J]. **Lecture Notes in Computer Science**, 2009, **5789 LNCS**:355-370.

[6] LI Aiping, TAN Shuang, JIA Yan. A method for achieving provable data integrity in cloud computing [J]. **Journal of Supercomputing**, 2016, **72**(1):1-7.

[7] ESCALA A, HEROLD G, KILTZ E, *et al*. An

algebraic framework for Diffie-Hellman assumptions [J]. **Lecture Notes in Computer Science**, 2013, **8043 LNCS**(PART 2):129-147.

[8] HUSSIEN Z A, JIN Hai, ABDULJABBAR A, *et al*. Public auditing for secure data storage in cloud through a third party auditor using modern ciphertext [C] // **Proceedings of the 2015 11th International Conference on Information Assurance and Security, IAS 2015**. Marrakesh: IEEE Press, 2015:73-78.

[9] ZHANG Jianhong, MENG Hongxin, YU Yong. Achieving public verifiability and data dynamics for cloud data in the standard model [J]. **Cluster Computing**, 2017, **20**(3):2641-2653.

[10] 李　勇,姚　戈,雷丽楠,等. 基于多分支路径树的云存储数据完整性验证机制[J]. 清华大学学报(自然科学版), 2016, **56**(5):504-510.
LI Yong, YAO Ge, LEI Linan, *et al*. LBT-based cloud data integrity verification scheme [J]. **Journal of Tsinghua University (Science and Technology)**, 2016, **56**(5):504-510. (in Chinese)

[11] ZHANG Yihua, BLANTON M. Efficient dynamic provable possession of remote data via update trees [C] // **ASIA CCS 2013 - Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security**. NY: ACM, 2013.

[12] SATO F, WAKABAYASHI S. Bloom filters based on the B-tree [C] // **Proceedings of the International Conference on Complex, Intelligent and Software Intensive Systems, CISIS 2009**. Fukuoka: IEEE Computer Society, 2009:500-505.

[13] SHIMBRE N, DESHPANDE P. Enhancing distributed data storage security for cloud computing using TPA and AES algorithm [C] // **Proceedings — 1st International Conference on Computing, Communication, Control and Automation, ICCUBEA 2015**. Piscataway: IEEE, 2015:35-39.

[14] BRODER A, MITZENMACHER M. Network applications of Bloom filters: a survey [J]. **Internet Mathematics**, 2004, **1**(4):485-509.

Dynamic Bloom filter based cloud storage data possession verification method

XIE　Lixia*, HU　Lijie

(School of Computer Science and Technology, Civil Aviation University of China, Tianjin 300300, China)

Abstract: In order to solve the inefficient verification problem of provable data possession in cloud storage, a cloud storage data possession verification method is proposed based on dynamic Bloom filter. Firstly, the homomorphic Hash function is used to process the cloud storage data, which simplifies the calculation of the client. Then, the third party verification platform uses the data block tags to construct the dynamic Bloom filter, which can support full dynamic operation. Finally, the ownership proof is generated by the random verification path, which enhances the cloud storage data possession verification security. The experimental results show that the proposed method can reduce the proof calculation overhead, enhance the verification security and efficiency.

Key words: cloud storage; possession verification; dynamic Bloom filter; dynamic operation; verification path