BTDA:基于半可信第三方的动态云数据更新审计方案

金 瑜[1] 蔡 超[1,2] 何 亨[1,2] 李 鹏[1,2]
(武汉科技大学计算机科学与技术学院 武汉 430065)[1]
(湖北省智能信息处理与实时工业系统重点实验室 武汉 430065)[2]

摘 要 云存储由于具有方便和廉价的优点,自诞生以来便得到了广泛应用。但与传统系统相比,云存储中的用户失去了对数据的直接控制,因此用户最关心的是存储在云上的数据是否安全,其中完整性是安全需求之一。公共审计是验证云数据完整性的有效方法。虽然现有方案不仅能够实现云数据的完整性验证,也能够支持动态数据更新审计,但它们也存在缺点,例如在执行多个二级文件块更新任务时,用户需要一直在线进行更新审计,而且在该过程中用户与云服务器的通信量和用户计算量都较大。基于此,提出了一种基于半可信第三方的动态云数据更新审计方案——BTDA。在BTDA中,用户将二级文件块更新审计任务代理给半可信第三方,因此在二级文件块更新审计过程中,用户可以离线,从而减少了用户端的通信量和计算量。另外,BTDA采用了数据盲化和代理重签名技术来防止半可信第三方和云服务器获取用户敏感数据,从而保护了用户隐私。实验表明,与目前的二级文件块更新审计方案相比,BTDA中的用户端无论在计算时间还是通信量方面都有大幅减少。

关键词 云存储,数据完整性,半可信第三方,二级文件块更新
中图法分类号 TP393.08 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.03.023

BTDA:Dynamic Cloud Data Updating Audit Scheme Based on Semi-trusted Third Party

JIN Yu[1] CAI Chao[1,2] HE Heng[1,2] LI Peng[1,2]
(College of Computer Science & Technology, Wuhan University of Science & Technology, Wuhan 430065, China)[1]
(Hubei Province Key Laboratory of Intelligent Information Processing and Real-time Industrial System, Wuhan 430065, China)[2]

Abstract Cloud storage has been widely used since its birth because of its convenience and low price. But compared with the traditional system, the users in the cloud storage system lost the direct control of the data, so users are most concerned about whether the data stored in the cloud are security, where integrity is one of the security needs. Public audit is an effective way to verify the integrity of cloud data. Existing research work can not only achieve cloud data integrity verification, but also support dynamic data update audit. However, such schemes also suffer from some drawbacks, for example, when multiple second-level file block update tasks are implemented, users need to be online for the update audit of each task, and in this process the communication cost and the computational cost on user side are larger. On this basis, this paper proposed BTDA, namely a semi-trusted third party dynamic cloud data update audit program. In BTDA, semi-trusted third party deals with update audit instead of user, so during the update audit process, the user can be off-line, thereby reducing the communication cost and the computational cost on user side. In addition, BTDA uses data blind and proxy re-signature technology to prevent semi-trusted third party and cloud server to obtain user sensitive data, thus protecting user privacy. Experiments show that compared with the current scheme about second-level file block update, BTDA has a large reduction in both computation time and communication cost on user side.

Keywords Cloud storage, Data integrity, Semi-trusted third party, Second-level file block update

1 引言

云存储是在云计算(cloud computing)上延伸和发展出来的一个新概念,是一种新兴网络存储技术。当云计算系统的运算和处理核心是大量数据的存储和管理时,该系统就转变成了一个云存储系统。简单来说,云存储就是将储存资源放到云上供人存取的一种新兴方案。使用者可以在任何时间、任何地点,通过任何连网装置连接到云上方便地存取数据,但

到稿日期:2016-12-22 返修日期:2017-04-11 本文受云计算环境下基于行为的动态信任模型研究(61303117),混合云中基于属性密码的轻量级安全控制机制研究(61602351),混合车载网络环境下的节点部署及协同内容分发研究(61502359)资助。

金 瑜(1973-),女,博士,副教授,主要研究方向为云计算、对等计算和信任模型,E-mail:973427045@qq.com(通信作者);蔡 超(1993-),男,硕士生,主要研究方向为云计算安全;何 亨(1985-),博士,副教授,主要研究方向为密码学、属性加密、安全控制机制;李 鹏(1978-),博士,副教授,主要研究方向为混合车载网络环境下的节点部署及协同内容分发。

是云存储仍然存在许多缺点和争议。数据安全和隐私是其中一个主要问题。与传统系统相比，云存储中的用户失去了对数据的直接控制，因此用户最关心的是存储在云上的数据是否完整。数据审计是解决这个问题的有效方法，其中最典型的就是公共审计。

虽然现有方案不仅能够远程实现云服务器上数据的完整性验证，也能够支持数据的动态更新，但是它们也存在一些缺点：由于批量更新数据块是文件中的常见操作，在执行多个二级文件块更新任务时，用户需要在线对每个任务进行更新审计，而且在该过程中用户与云服务器的通信量和用户计算量都较大。基于此，提出一种基于半可信第三方的云数据更新审计方案。该方案使用了数据盲化技术以及代理重签名技术，将二级文件块更新审计任务交给半可信第三方来进行，因此在二级文件块更新审计过程中，用户可以离线，从而减少了用户端的通信量和计算量。

2　相关工作

如何保证云服务器中存储的用户数据没有被修改、删除或者泄露，成为一个亟需解决的重要安全性问题。Deswarte 等[1]利用哈希函数来验证云服务器所存储的文件的完整性，允许客户端使用相同元数据进行多次挑战；但是该方案的缺点在于：云服务器要对整个数据文件取幂，并且需要访问所有文件块，计算复杂度大，效率低。2004 年，Sebe 等[2]提出了基于 Diffie-Hellman 问题的文件完整性检查方案；但该方案的缺点是用户存储的花销较大，并且也需要云服务器在产生证明的过程中访问所有文件。

Ateniese 等[3]于 2007 年针对不可信存储者，首次提出了数据拥有证明的公开审计概念(PDP)。通过云服务器对数据文件块进行随机选取，利用基于 RSA 的同态验证技术对存储在云服务器端的用户数据进行审计；但是，该方案仅适用于静态数据，不支持数据的动态操作。Juels 等[4]提出了“可回取证明模型”，利用隐藏在正常存储文件中的附加数据块(也称作“哨兵块”)检测云服务器端的数据是否完好地存储；但是，此方案可审计的次数需要预先设定，而且其不支持公开审计。

Shacham 等[5]于 2008 年设计了可证明安全的基于 BLS 短签名[6]的高效数据回取方案；但该方案没有考虑到隐私保护问题，有可能造成重要隐私数据被泄露。同年，动态数据可回取证明方案也随着云计算的发展得到了越来越多专家、学者的关注。例如，Shacham 等人[7]充分考虑了云数据的动态操作；随后，Wang 等[8]利用 Merkle Hash 树(MHT)来支持动态数据的操作。但是，以上方案均存在隐私泄露问题。

Wang 等[9]于 2010 年通过使用随机掩码来保护用户数据的隐私，但是该方案不支持数据的动态操作。Zhu 等[20]于 2011 年通过利用授权的第三方应用端(AA)来代理用户执行动态操作，即数据的更新审计；但应用端(AA)拥有用户的私钥，这样会导致方案存在单点失效与私钥泄露的风险。Wang 等[14]于 2013 年引入了安全的第三方来代替用户进行数据的签名，并且利用数据盲化技术保证用户数据的隐私，从而在公共审计的时候保护了共享数据的隐私；同年，Yang 等人[10]利用双线性对的性质对隐私数据进行加密，保证了用户数据隐私；并且采用了数据碎片技术，对数据块再进行分块，使得一个数据块包含多个二级文件块，从而减少了签名数量；但是该方案没有考虑到二级文件块的更新问题，会导致在更新过程中浪费存储空间。2014 年，Liu 等[11]首次提出了授权公共审计以及二级文件块更新方案(简称 FUPA)。FUPA 通过授权操作，消除了未授权审计方攻击云服务器的威胁；而且采取细粒化更新方法，解决了在大量二级文件块更新时浪费云服务器存储空间的问题。但是，FUPA 存在下列问题：1)在执行多个二级文件块的更新任务时，用户需要一直在线，对每个任务进行更新审计；2)在整个更新过程中用户与云服务器之间的通信量以及用户计算量都较大。

Wang 等[15]于 2015 年利用代理重签名技术有效解决了共享数据中群用户退出的问题，将退出用户的数据签名转变为原始用户的签名。

近年来，利用第三方审计来实现云存储安全的方法很多。Ranjith[16]和 He[17]利用第三方审计者来验证多个云服务器上所存储的数据的完整性，并且当数据损坏时能够定位损坏数据的位置，从而进行数据修复。He[18]在同一个框架上实现了重复数据删除和数据完整性检查；云服务器可以正确地检查新所有者的身份，并且第三方审计者可以验证删除重复数据后数据的完整性。Shen[19]通过引入第三方中介者代替用户进行数据的签名以及数据审计，减轻了用户的计算负担并且保护了用户的隐私。但是这些方案只是利用第三方对用户数据进行完整性审计或者利用第三方代替用户进行数据签名，不能够代替用户进行数据的动态操作，包括数据更新以及更新验证(统称为数据的更新审计)。

基于 FUPA 的不足，本文应用文献[14-15]方案中的数据盲化技术以及代理重签名技术，引入半可信第三方，提出了 BTDA：一种基于半可信第三方的云数据更新审计方案。BTDA 具有以下优点：1)由于采用半可信第三方代替用户进行二级文件块的更新审计，因此用户在更新过程中可以离线；2)与 FUPA 相比，BTDA 在整个更新过程中产生的通信量以及用户半可信第三方的计算量都有所减少。

3　BTDA

3.1　系统模型

BTDA 的系统模型如图 1 所示。系统中包含 4 个实体：用户(User)、云服务器(Cloud Server)、半可信第三方(Third Party)，以及公共审计方(TPA)。方案的大致过程如下：

1)用户初始化，将数据、签名集合等上传到云服务器上进行存储；

2)执行多个更新任务时，用户将盲化数据集合 *ReqTp* 发送给半可信第三方，半可信第三方产生更新审计队列；

3)用户将更新请求集合 *ReqCss* 发送给云服务器，云服务器构造更新请求队列，并依次开始更新数据；

4)云服务器将每一次的更新证据 *UpdateProof* 发送给半可信第三方；

5)半可信第三方根据更新审计队列及 *UpdateProof* 对每一个更新任务进行更新审计，判断云服务器是否更新成功并保存更新审计结果，若成功则计算新的签名集合 *Tag* 并发送给云服务器，云服务器进行代理重签名并更新签名，然后执行下一个更新请求；

6)半可信第三方将所有任务的更新审计结果返回给用户;

7)验证数据的完整性,用户向公共审计方发送验证请求 *VerifyRequest*;

8)公共审计方收到验证请求后,向云服务器发送挑战;

9)云服务器收到挑战后产生完整性验证证据 *Response*,并发送给公共审计方;

10)公共审计方根据 *Response* 验证用户数据的完整性,并将验证结果 *VerifyResult* 发送给用户。

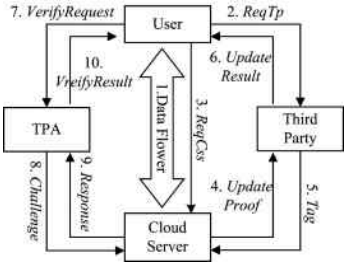

图 1 BTDA 系统模型图

Fig. 1 System model diagram of BTDA

3.2 威胁模型与安全目标

在 BTDA 中,主要考虑 3 种攻击:

1)Cloud Server 是半可信的,即云服务器如实存储用户数据,但在数据丢失和更新失败等情况下,为了维护自己的名声或者避免损失金钱,可能会欺骗公共审计方以及 Third Party;

2)Third Party 是半可信的,即如实更新审计任务,但可能会窃取用户数据,导致用户数据隐私外泄;

3)TPA 也是半可信的,即如实执行公共审计任务,但可能会在审计过程中窃取用户数据隐私。

为了应对上述攻击,方案要达到以下安全目标:

1)执行公共审计,即公共审计方能够代表用户正确地验证数据的完整性;

2)执行更新审计,即半可信第三方能够代表用户验证云服务器是否正确更新了数据;

3)保护数据隐私,即在公共审计和更新审计过程中,公共审计方和半可信第三方都不能够窃取用户数据隐私。

3.3 预备知识

3.3.1 双线性图

假设 G_1,G_2,G_T 都是素数阶为 p 的乘法循环群,g_1 和 g_2 分别是 G_1 和 G_2 的生成元;一个双线性图的构造方法是:e:$G_1\times G_2\rightarrow G_T$。那么 e 具有以下几个性质:

1)双线性:$\forall m\in G_1, n\in G_2\Rightarrow e(m^a,n^b)=e(m,n)^{ab}$;

2)非退化性:$e(g_1,g_2)\neq 1$;

3)可计算性:e 能够被有效地计算。

3.3.2 带权值的梅肯哈希树(RMHT)

RMHT 的详细构造过程如图 2 所示。RMHT 的节点由 $\{\kappa,r_N\}$ 组成,其中 κ 为哈希值,r_n 为节点的权值。首先,RMHT 的叶子节点 L_i 是以数据块 m_i 为基础构造而成的,m_i 包含 s_i 个二级文件块,表示为 $\{m_{i,1},\cdots,m_{i,s_i}\}$;利用图 3 所示的 L_i 构造树,可以求得叶子节点 L_i 的哈希值为 $\kappa=h(root_i)$,$r_{L_i}=s_i$,因此可以计算出叶子节点 $L_i=\{h(root_i),s_i\}$,然后以子节点构造父节点。假设父节点为 N_p,其有 2 个子节点 $N_1=\{\kappa_1,r_{N_1}\}$,$N_2=\{\kappa_2,r_{N_2}\}$,则 $N_p=\{h(\kappa_1\parallel\kappa_2),(r_{N_1}+r_{N_2})\}$,其中 $\parallel$ 为串联操作。在应用中还用到哈希树辅助信息的概念,如图 2 所示,当知道叶子节点 $root_2$ 时需要辅助信息 $\Omega_2=(h(root_1),h(e),h(c))$ 获取哈希树的根 R,Ω_i 即为 RMHT 中相应叶子节点的辅助信息;而在图 3 的构造叶子节点 L_i 的构造树中,$m_{i,2}$ 的辅助信息为 $\Omega_{i,2}=(h(m_{i,1}),h(b))$。

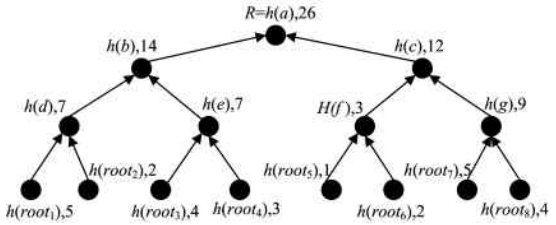

图 2 RMHT 的构造过程

Fig. 2 Construction process of RMHT

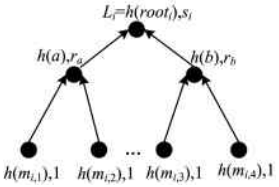

图 3 RMHT 中叶子节点 L_i 的构造过程(L_i 构造树)

Fig. 3 Construction process of leaf nodes L_i in RMHT (L_i construction tree)

3.3.3 代理重签名

代理重签名由 Blaze 等[12]首次提出。该方案使得一个半信任的代理方能够把两个用户的签名互相转换,也就是代理方能够将 Alice 对数据块的签名转化为 Bob 对该数据块的签名;并且在代理重签名的过程中,代理方不能获取两个用户的私钥。

3.3.4 数据盲化

Chaum 等[13]提出了盲签名方案,其提供了一种数据盲化方法。用户 A 将数据 m_i 盲化处理为 $m_i'=\mu^{m_i}*g_1^r$,其中 μ,$g_1\in G_1$,随机产生 $r\in Z_p$;用户 B 获取到盲化数据 m_i' 时,由于在 G_1 群中的离散对数问题,用户 B 不能通过 μ^{m_i} 获取 m_i;并且由于存在随机值 g_1^r,因此不能通过盲化数据组合操作推测出任何有用信息。

3.3.5 二级文件块更新操作类型的定义

1)PM:部分更改,即一个数据块中的一个二级文件块需要修改;

2)PD:部分删除,即一个数据块中的一个二级文件块需要被删除;

3)PI:部分插入,即一个数据块中需要插入一个二级文件块。

3.4 BTDA 详述

BTDA 由初始化、数据更新以及更新审计、公共审计 3 个部分组成。

3.4.1 初始化

使 G_1,G_2,G_T 都是以 p 为素数阶的乘法循环群,双线性图 e:$G_1\times G_2\rightarrow G_T$;$g_1$,$g_2$ 分别是 G_1,G_2 的生成元。其中,H:$\{0,1\}^*\rightarrow G_1$ 是安全哈希函数,能够将信息摘要 M_{mfo} 映射到 G_1 循环群中。产生了基本参数后,客户端开始执行以下算法。

$KeyGen(1^k)\rightarrow(PK,\alpha)$:客户端产生一个秘密值 $\alpha\in Z_p$,

并且在 G_2 中产生一个生成元 g_2，在 G_1 中产生一个生成元 g_1，然后计算 $v=g_2^{\alpha}$。其中私钥 $Sk=\{\alpha\}$；公钥 $Pk=\{g_1,g_2,v\}$。

$TagGen(F,\alpha,SegReq)$：根据分块要求 $SegReq$（包含 $s_{\max}$，即每个数据块能分为二级文件块数量的上限）；客户端将文件 F 划分为 $F=\{m_{i,j}\}$，$i\in[1,l]$，$j\in[1,s]$，$s_i\in[1,s_{\max}]$。其中 F 被分为 l 个数据块，第 i 块又被划分为 s_i 个二级文件块。通过 L_i 构造树 T_{L_i} 可以得到 $L_i=\{H(root_i),s_i\}$，客户端利用 L_i 构造一个 RMHT（用 T 表示，R 为 T 的根节点）。然后对根节点进行签名：$sigR=(H(R))^{\alpha}$。$x_1,q\in Z_p$ 为随机选取的值，生成 n 个数 $\{x_1,x_2,\cdots,x_n\in Z_p\}$，$n\in[1,s_{\max}]$ 是以 q 为公比的等比数列；而对于每一个数据块 $m_i(i\in[1,l])$，可以计算它的签名：$\sigma_i=[H(root_i)*\prod_{j=1}^{s}\mu_j^{m_{i,j}}]^{\alpha}$，$\mu_j=g_1^{x_j}\in G_1(x_1,x_2,\cdots,\in Z_p)$；可以用 $\phi=\{\sigma_i\}(i\in[1,l])$ 表示签名集合，用 $T_l=\{T_{L_1},\cdots,T_{L_l}\}$ 表示所有的 L_i 构造树集合，$X=\{x_1,x_2,\cdots,x_n\in Z_p\}$，$n\in[1,s_{\max}]$，$U=\{\mu_j\in Z_p\}$，$j\in[1,s_{\max}]$。最后，算法将 $\{F,\phi,T,R,sigR,U,q,X,T_L\}$ 作为输出。用户将 $\{F,\phi,T,R,sigR,U,T_L\}$ 发送给云服务器，将 $\{q,X\}$ 发给半可信第三方。

3.4.2　数据更新以及更新审计

$ReadyUpdate()$：对于每一个更新请求 $update=(i,k,o,m')$（i 表示数据块号，k 表示数据块中二级文件块的块号，o 表示更新操作，m' 表示新的数据），用户随机选取 $r\in Z_p$，则 $requestCss=\{i,k,o,m',r\}$；并且计算 $Pk_r=(g_1^{-r})^{\alpha}$，m' 对应盲化值以及哈希值，从而计算出 $requestTp=\{i,k,o,MP=\mu_k^{m'}*g_1^{-r},H(m'),Pk_r\}$，当 o 为 PD 操作时，MP 作为零处理。假设进行批量更新时有 w 个更新任务，用 $ReqCss=\{requestCss_i\}(i\in[1,w])$ 表示更新请求集合，$ReqTp=\{requestTp_i\}(i\in[1,w])$ 表示致盲数据集合。最后，算法将 $ReqTp$ 和 $ReqCss$ 作为输出。用户将 $ReqTp$ 和 $ReqCss$ 分别发送给半可信第三方和云服务器。如图 4 所示，根据这两个集合，云服务器和半可信第三方分别建立请求更新请求队列和更新审计队列。随后，云服务器开始执行 $ReKey$ 算法。

$ReKey(\alpha,\beta)\rightarrow rk_{\beta\rightarrow\alpha}$：云服务器获取代理签名密钥，分为以下 4 步：

1）云服务器作为代理产生一个随机值 $x\in Z_p$，并且将 x 发送给客户端；

2）客户端收到后，将 $x*\alpha$ 发送给半可信第三方；

3）半可信第三方收到后，发送 $x*\alpha/\beta$ 给云服务器；

4）云服务器计算出代理重签名密钥 $rk_{\beta\rightarrow\alpha}=\alpha/\beta$。

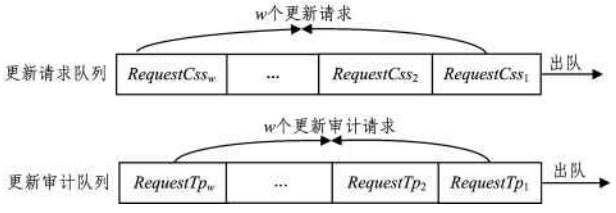

图 4　队列示意图

Fig. 4　Queue diagram

当这些准备工作都完成后，云服务器进行数据更新，接下来将在服务器和半可信第三方之间进行更新审计，过程如下。

1）云服务器从更新队列中取出一个更新请求 $requestCss$，执行更新算法。

$PerformUpdate(requestCss)$：云服务器根据更新请求 $requestCss=\{i,k,o,m',r\}$，将数据块 m_i 更新为 m_i'（二级文件块数由 $s_i\rightarrow s_i'$），根据图 3 中 L_i 构造树的过程，重新构造 T'_{L_i}（与文献[8]中的 MHT 变化一样），重新计算 L_i 的值 L_i'；并且根据叶子节点 L_i'，按照 RMHT 的构造过程生成新的 RMHT，用 T' 表示，T' 的根节点用 R' 表示。根据数据块 m_i 和 m_i' 计算下列值：当 $o=$ PM 时，$M_1=\prod_{j=1}^{k-1}\mu_j^{m_{i,j}}*g_1^{-r}$，$M_2=\mu_k^{m_{i,k}}*g_1^{-r}$，$M_3=\prod_{k+1}^{s'}\mu_j^{m'_{i,j}}*g_1^{-r}$；当 $o=$ PD 时：$M_1=\prod_{j=1}^{k-1}\mu_j^{m_{i,j}}*g_1^{-r}$，$M_2=\mu_k^{m_{i,k}}*g_1^{-r}$，$M_3=\prod_{k}^{s'}\mu_j^{m'_{i,j}}*g_1^{-r}$；当 $o=$PI 时，$M_1=\prod_{j=1}^{k-1}\mu_j^{m'_{i,j}}*g_1^{-r}$，$M_2=\mu_k^{m'_{i,k}}*g_1^{-r}$，$M_3=\prod_{k+2}^{s'}\mu_k^{m'_{i,k}}*g_1^{-r}$。最后输出 $UpdateProof:\{\Omega_{i,k},\Omega_i,\sigma_i,H(m_{i,k}),sigR,R',M_1,M_2,M_3\}$。

2）收到云服务器发送过来的更新证据后，半可信第三方从更新审计队列中取出 $requestTp$，并根据自己的私钥 $skt=\beta\in Z_p$ 以及公钥 $Pkt=g_2^{\beta}\in G_2$ 执行验证算法。

$Verify(requestTp,UpdateProof)$：首先根据 $(\Omega_{i,k},H(m_{i,k}))$ 计算出 T 的叶子节点 L_i 的哈希值 $H(root_i)$，再根据 $(\Omega_i,H(root_i))$ 计算出 T 的根节点 R，验证 $sigR$ 的正确性；如果 $e(H(R),v)?=e(sigR,g_2)$ 不相等则输出 false，否则进行下一步验证；根据不同的操作计算出新的根节点 $R''\leftarrow(H(root_i'),\Omega_i)$，其中 $H(root_i')$ 是 L_i 构造树执行更新操作后新根节点的哈希值，$H(root_i')\leftarrow(H(m'),\Omega_{i,k})$；验证 R'' 是否与 R' 相等。若验证成功，则通过下面的公式验证 M_1,M_2,M_3 的正确性。

$$\text{PM}:e(H(root_i)*\prod_{j=1}^{3}M_j,v)?=e(\sigma_i*\prod_{j=1}^{3}Pk_r,g_2)$$

$$\text{PD}:e(H(root_i)*\prod_{j=1}^{2}M_j*M_3^q,v)?=e(\sigma_i*\prod_{j=1}^{2}Pk_r*Pk_r^q,g_2)$$

$$\text{PI}:e(H(root_i)*\prod_{j=1}^{2}M_j*M_3^{\frac{1}{q}},v)?=e(\sigma_i*\prod_{j=1}^{2}Pk_r*Pk_r^{\frac{1}{q}},g_2)$$

验证通过后，开始计算根节点 R' 的签名，并且根据不同操作计算更新后数据的签名：$sigR'=(H(R))^{\beta}$，并验证以下公式的正确性。

$$\text{PM}:\sigma_i'=(H(root_i')*M_1*MP*M_3)^{\beta}$$

$$\text{PD}:\sigma_i'=(H(root_i')*M_1*M_3)^{\beta}$$

$$\text{PI}:\sigma_i'=(H(root_i')*M_1*MP*M_2^q*M_3)^{\beta}$$

若验证成功，则输出 $Tag=(sigR',\sigma_i',Pkt,Pkt^q)$，并保存验证结果。

3）半可信第三方将 Tag 发送给云服务器，云服务器收到返回的数据后执行代理重签名算法。$ReSign(\sigma_i',sigR',Pkt,Pkt^q,r,sk_{\beta\rightarrow\alpha})$：首先对 $sigR'$ 进行代理重签名，$sigR_{\text{new}}=(sigR')^{sk_{\beta\rightarrow\alpha}}=(H(R'))^{\alpha}$；然后根据不同操作对 σ_i' 进行处理：

$$\text{PD}:\sigma''=\sigma_i'*Pkt^r*Pkt^r=(H(root_i')*\prod_{j=1}^{s}\mu_j^{m'_{i,j}})^{\beta}$$

$$\text{PM}:\sigma''=\sigma_i'*Pkt^r*Pkt^r*Pkt^r=(H(root_i')*\prod_{j=1}^{s}\mu_j^{m'_{i,j}})^{\beta}$$

$$\text{PI}:\sigma''=\sigma_i'*Pkt^r*Pkt^r*Pkt^r*Pkt^{q*r}=(H(root_i')*\prod_{j=1}^{s}\mu_j^{m'_{i,j}})^{\beta}$$

上述计算完成后，对 σ_i'' 进行代理重签名，$\sigma_{\text{new}}=(\sigma'')^{sk_{\beta\rightarrow\alpha}}=[H(root_i')*\prod_{j=1}^{s'}\mu_j^{m'_{i,j}}]^{\alpha}$，随后云服务器更新相应的签名。

4)至此,一次更新以及更新审计过程完成,随后依次执行队列中的每一个更新任务,直到所有队列中的任务全部执行完毕。半可信第三方将所有更新审计结果报告给用户。PM更新审计的流程如图5所示。

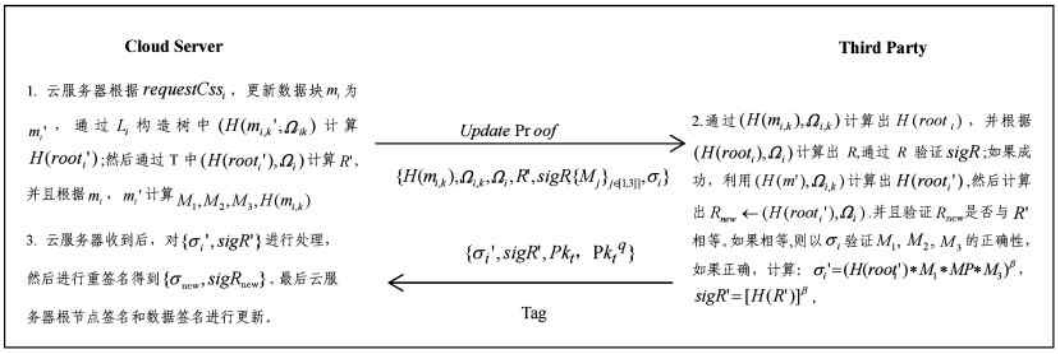

图5 PM操作时的数据更新审计过程

Fig. 5 Data update audit process during PM operation

3.4.3 公共审计

该过程与文献[10]中的过程类似,即利用双线性对的性质保护数据隐私。公共审计方(TPA)从[1,n]中随机选取 c 个元素的子集 $I=\{ss_1,ss_2,ss_3,\cdots,ss_c\}$(假设 $ss_1\leqslant ss_2\leqslant ss_3\leqslant ss_4\leqslant\cdots\leqslant ss_c$)。对于每一个 $ss_i\in I$,TPA 会相应地选取 $vv_i\in Z_p$,然后随机选取 $\eta\in Z_p$,计算 $\varepsilon=(v)^\eta$,因此 TPA 产生的挑战可以表示为:$chal=(\{vv_i,i\}_{i\in I},\varepsilon)$,此 $chal$ 被发送给云服务器。

云服务器收到 TPA 的挑战 $chal$ 后,执行证据产生算法。

$GenProof(F,\phi,chal)$:云服务器根据收到的 $chal$ 计算 $\sigma=\prod_{i\in I}\sigma_i^{vv_i}$,$TP_j=\sum_{i\in I}vv_i*m_{i,j}$,$j\in[1,\tilde{\omega}]$,$u=\prod_{j=1}^{\tilde{\omega}}e(\mu_j,\varepsilon)^{TP_j}$,$\tilde{\omega}=\max\{s_i\}_{i\in I}$;并且返回 $Response=\{u,\sigma,(H(root_i),\Omega_i)_{i\in I},sigR\}$,云服务器将证明 $Response$ 发送到 TPA。

TPA 收到云服务器的证明后,执行验证算法。

$VerifyProof(v,Response,chal)$:TPA 通过 $\{H(root_i),\Omega_i\}_{i\in I}$ 计算出 RMHT 的根节点 R,通过判断 $e(sigR,g_2)$ 和 $e(H(R),v)$ 是否相等来验证 $sigR$。$sigR$ 验证成功后,首先计算 $H_{chal}=\prod_{i\in I}[H(root_i)]^{\eta*vv_i}$,然后验证等式 $u*e(H_{chal},v)=(\sigma,g_2^\eta)$ 是否成立,若成立,则数据完整。

4 安全性分析

与 FUPA 相比,本文方案 BTDA 的安全性并没有降低。下面分3个部分进行安全性分析,从而证明方案的可靠性。

1)公共审计过程中的安全性

定理 1 在公共审计过程中,云服务器返回正确的证据,公共审计方审计时能够通过验证。

证明:如果云服务器的证据正确,则 $u*e(H_{chal},v)=e(\sigma,g_2^\eta)$ 成立。证明过程与文献[10]中的过程类似,这里省略。

定理 2 在公共审计过程中,云服务器不能够伪造数据签名来通过 TPA 的审计。

证明:CSP 在已知用户的元数据、数据块签名和公共参数 g_2 和 v 的情况下计算伪造一个新的数据块签名,是一个 Diffie-Hellman 难题。Diffie-Hellman 难题可以描述为:在知道 g_2,g_2^a,g_2^b 的情况下计算 g_2^{ab} 是困难的,因为很难求出 a 和 b 的值。

2)动态更新审计过程的安全性

定理 3 云服务器不能够伪造动态更新后的证据来通过半可信第三方的更新审计。

证明:如果云服务器要伪造更新后的证据来通过半可信第三方的更新审计,首先,必须伪造旧的 RMHT 的根 R 的签名,这需要云服务提供者获得 α,但根据 DHP(Diffie-Hellman problem),CSP 在知道公共参数 v 和 g_2 的情况下要计算出密钥 α 是困难的;其次,云服务器必须提供正确的 M_1,M_2,M_3,以使得下面的公式成立(以 PM 操作为例,其他操作证明过程类似)。

$$\text{PM}:e(H(root_i)*\prod_{j=1}^{3}M_j,v)?=e(\sigma_i*\prod_{j=1}^{3}Pk_r,g_2)$$

其推导过程如下:

$$\begin{aligned}\text{左边}&=e(H(root_i)*\prod_{j=1}^{s}\mu_j^{m_{i,j}}*\prod_{j=1}^{3}g_1^{-r},v)\\&=e([H(root_i)*\prod_{j=1}^{s_i}\mu_j^{m_{i,j}}]^\alpha*\prod_{j=1}^{3}g_1^{-r*\alpha},g_2)\\&=e(\sigma_i*\prod_{j=1}^{3}Pk_r,g_2)\\&=\text{右边}\end{aligned}$$

3)用户的隐私安全

定理 4 公共审计方不能够根据已收到的数据获取到用户的私密数据。

证明:根据密码学中哈希函数的单向性,TPA 不能够从叶子节点哈希值中提取到文件块的信息;同时,本文采取了文献[10]的方法,即利用双线性对的性质,防止 TPA 通过数据块的线性组合推导出用户的数据隐私信息。

定理 5 半可信第三方不能够根据收到的动态更新证据获取到用户的私密数据。

证明:云服务器将 M_1,M_2,M_3 等盲化的数据发送给半可信第三方,基于盲化数据的特性:$m_i'=\mu^{m_i}*g_1^r(\mu,g_1\in G_1)$,随机产生 $r\in Z_p$。半可信第三方获取到盲化数据 m_i' 时,由于在 G_1 群中的离散对数问题,用户 B 不能通过 μ^{m_i} 获取 m_i,并且由于存在随机值 g_1^r,用户 B 也不能通过盲化数据组合操作推测到任何有用信息。

5 性能分析

本文介绍的 BTDA 与其他相关方案的比较结果如表1所列。

表 1　相关方案与本文方案的比较

Table 1　Comparison of related schemes with this paper

	文献[1-4]	文献[5-10]，文献[14-19]	文献[20]	FUPA	BTDA
私有审计	√	√	√	√	√
第三方完整性审计	×	√	√	√	√
第三方更新审计	×	×	√	×	√
二级文件块更新操作	×	×	×	√	√

可以看出，相比于 FUPA，BTDA 能够进行第三方更新审计，代替用户进行数据的动态操作。BTDA 与文献[20]一样拥有第三方数据更新审计功能；但是在 BTDA 中半可信第三方不拥有用户私钥，从而解决了其安全问题，并且支持第三方代替用户进行二级文件块的更新审计。文献[20]采用索引哈希表的方式进行数据的更新，而 BTDA 采用的是梅肯哈希树的方式。理论上，MHT 动态操作的复杂度为 $O(\log(n))$，索引哈希表插入和删除的复杂度为 $O(n)$；实验也能够证明梅肯哈希树的方式在数据更新时更高效。

在实验中，客户端、云服务器以及半可信第三方的运行环境如下。

1）软件环境：操作系统：32 位 Windows7；开发软件：Eclipse；编程语言：Java；Java Pairing-Based Cryptography（JPBC）：jpbc-2.0.0。采用密码学 API 完成方案中的算法。

2）硬件环境：CPU：Intel Core i3 2.10GHz，RAM：2.00GB。

以 $\lambda=80$bit 为安全参数，因此 G_1，G_2 中的生成元 $|p|=160$bit，二级文件块的大小为 20Byte。

本实验的云审计系统由 5 台 PC 机组成。其中，3 台 PC 机用于搭建 Cassandra 云存储服务器，存储用户的数据和数据签名，并且在用户数据公共审计时提供证据；1 台 PC 机作为半可信第三方完成数据的动态更新审计；1 台 PC 机作为云用户。

实验 1　取数据块的个数 n 为 1000000，对整个数据分别进行 200 次插入、删除以及更改操作，比较梅肯哈希树和索引哈希表完成更新所花费的时间（如表 2 所列）。实验表明，在插入、删除和更改操作概率相同的情况下，梅肯哈希树的总体性能优于索引哈希表。

表 2　MHT 与 IHT 完成更新时间的比较

Table 2　Comparision of update time between MHT and IHT

（单位：ms）

	MHT 完成更新时间	IHT 完成更新时间
插入	30	65
删除	22	66
更改	30	4

实验 2　在更新次数发生变化时，考查用户端所花费的时间。在 FUPA 中用户端花费的时间从提交更新请求开始，到云服务器收到所有更新后的数据块签名时结束，由用户端计算时间和用户端与云服务器通信的时间组成；在 BTDA 中用户端花费的时间，即用户端提交所有的更新请求到云服务器上以及提交盲化数据到半可信第三方的时间，由用户端与云服务器通信的时间以及用户端与半可信第三方通信的时间组成。在实验中，选择数据大小为 10kB，分为 10 个数据块；每个数据块分为 50 个二级文件块，在第 i 块中插入大小为 20Byte 的新数据以进行更新。图 6 给出了 BTDA 方案与 FUPA 方案[11]的比较结果。从图中可以看出，BTDA 在进行批量数据更新时，用户所花费的时间比 FUPA 要少得多。这是因为用户不用一直在线对每一个更新任务进行更新审计，只需要将计算的盲化数据发给半可信第三方，让其代替自己进行数据的更新审计。

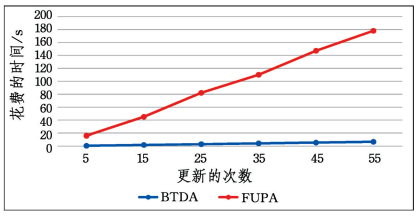

图 6　用户执行更新操作所消耗的时间与更新次数的关系

Fig. 6　Relationship between the time consumed by the user in updating the operation and the number of updates

实验 3　在一次更新中，二级文件分块数发生变化时，考查半可信第三方和用户端总的计算时间。在 FUPA 中，总的计算时间由用户端验证更新结果的时间和计算新的数据块签名的时间组成。在 BTDA 中，总的计算时间由用户端计算致盲数据时间、半可信第三方验证更新结果的时间和计算新的数据块签名的时间组成。在实验中，选择数据大小 10kB，分为 10 个数据块，对第 i 块进行更新审计，在第 i 块中插入大小为 20Byte 的新数据以进行更新。实验结果如图 7 所示。从图 7 中可以看出，在 BTDA 中半可信第三方和用户端的总计算时间不会随着数据块划分的二级文件块数量的增加而增加，基本维持不变；而在 FUPA 中客户端的计算时间随着数据块划分的二级文件块数量的增加而呈线性增长。这是因为本文方案在计算新的签名时，不用计算 $\prod_{j=1}^{s}\mu_j^{m'_{ij}}$（$s$ 为二级文件块的数量）。

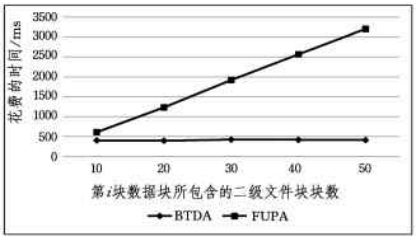

图 7　一次更新任务所花费的总时间与二级文件块划分块数的关系

Fig. 7　Relationship between total time spent on an update task and second-level file block number

实验 4　在二级文件分块数发生变化时，考查一次更新系统产生的通信量。实验数据大小为 10kB，分为 10 个数据块，在第 i 块中插入大小为 20Byte 的新数据以进行更新。在 BTDA 中一次更新产生的通信量包含 3 部分：1）用户端和半可信第三方之间的通信量；2）半可信第三方和云服务器之间的通信量；3）用户端与云服务器之间的通信量。而在 FUPA

中，一次更新产生的通信量仅发生在用户端和云服务器之间。实验结果如图 8 所示。从图 8 中可见，在 BTDA 中一次更新产生的通信量随着数据块划分的二级文件块数量的增加而面基本维持不变，而在 FUPA 中通信量随着数据块划分的二级文件块数量的增加而呈线性增长。其原因在于，BTDA 在更新审计时不需要回取整个数据块。

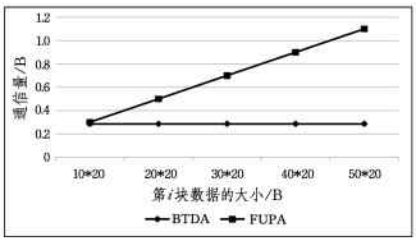

图 8 执行一次更新任务时产生的通信量与数据块大小的关系

Fig. 8 Relationship between the generated communication amount and the size of data blocks when performing an update

结束语 针对目前云数据动态更新方案中存在的不足，本文提出了一种基于半可信第三方的动态云数据更新审计方案——BTDA。在 BTDA 中，将用户端更新审计工作代理给半可信第三方，用户端可以离线，不用参与整个更新审计过程，从而大大减少了用户端的计算量；而且在更新过程中采用数据盲化和代理重签名技术，因而半可信第三方和云服务器无法获取到用户的敏感数据，保护了数据隐私；另外，在更新审计过程中，现有方案需要返回整个数据块，因而通信量较大，而在 BTDA 中云服务器并不需要返回整个数据块，只需返回 3 个固定大小的数据，从而大大减少了更新审计产生的通信量。实验表明：与 FUPA 方案相比，BTDA 无论在计算时间还是通信量方面都有较大减少。

参 考 文 献

[1] DESWARTE Y, QUISQUATER J J, SAÏDANE A. Remote Integrity Checking[M]// Integrity and Internal Control in Information System VI. 2004: 1-11.

[2] SEBE F, MARTINEZ-BALLESTE A, DESWARTE Y. Time-bounded remote file integrity checking: Technical Report 04429 [R]. 2004.

[3] ATENIESE G, BURNS R, CURTMOLA R, et al. Provable data possession at untrusted stores[C]// ACM Conference on Computer and Communications Security. ACM, 2007: 598-609.

[4] JUELS A, KALISKI B S. Pors: proofs of retrievability for large files[C]// ACM Conference on Computer and Communications Security. ACM, 2007: 584-597.

[5] SHACHAM H, WATERS B. Compact Proofs of Retrievability [M]// Advances in Cryptology-SIACRYPT 2008. Springer Berlin Heidelberg, 2008: 442-483.

[6] DAN B, LYNN B, SHACHAM H. Short Signatures from the Weil Pairing[J]. Journal of Cryptology, 2004, 17(4): 297-319.

[7] SHACHAM H, WATERS B. Compact Proofs of Retrievability [C]// Proc. 14th Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT'08). 2008: 90-107.

[8] WANG Q, WANG C, REN K, et al. Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing [J]. IEEE Transactions on Parallel & Distributed Systems, 2011, 22(5): 847-859.

[9] WANG C, WANG Q, REN K, et al. Privacy-preserving public auditing for data storage security in cloud computing[J]. Proceedings-IEEE INFOCOM, 2010, 62(2): 525-533.

[10] YANG K, JIA X. An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing[J]. IEEE Transactions on Parallel & Distributed Systems, 2013, 24(9): 1717-1726.

[11] LIU C, ZHANG X, YANG C, et al. Authorized Public Auditing of Dynamic Big Data Storage on Cloud with Efficient Verifiable Fine-grained Updates[J]. IEEE Transactions on Parallel & Distributed Systems, 2013, 25(9): 2234-2244.

[12] BLAZE M, BLEUMER G, STRAUSS M. Divertible protocols and atomic proxy cryptography[J]. Eurocrypt, 1998, 1403: 127-144.

[13] CHAUM D. Blind Signatures for Untraceable Payments[M]// Advances in Cryptology. Springer US, 1983: 199-203.

[14] WANG B, CHOW S S M, LI M, et al. Storing Shared Data on the Cloud via Security-Mediator[C]// IEEE International Conference on Distributed Computing Systems. IEEE Computer Society, 2013: 124-133.

[15] WANG B, LI B, LI H. Panda: Public Auditing for Shared Data with Efficient User Revocation in the Cloud[J]. IEEE Transactions on Services Computing, 2015, 8(1): 92-106.

[16] RANJITH G, CHANDRA J V, SAGARIKA P, et al. Intelligence based Authentication- Authorization and Auditing for secured data storage[J]. Journal of Advanced in Engineering and Technology, 2015, 8(4): 628-636.

[17] HE K, HUANG C H, ZHOU C H, et al. Public auditing for encryption data with client-side deduplication in cloud storage[J]. Wuhan University Journal of Natural Sciences, 2015, 20(4): 291-298.

[18] HE K, HUANG C H, SHI J L, et al. Public Integrity Auditing for Dynamic Regenerating Code Based Cloud Storage[C]// IEEE Symposium on Computers and Communication (ISCC). 2016: 581-588.

[19] SHEN W T, YU J, XIA H, et al. Light-weight and privacy-preserving secure cloud auditing scheme for group users via the third party medium[J]. Journal of Network and Computer Applications, 2017, 82: 56-64.

[20] ZHU Y, WANG H, HU Z, et al. Dynamic audit services for integrity verification of outsourced storages in clouds[C]// Proceedings of the 2011 ACM Symposium on Applied Computing. ACM, 2011: 1550-1557.