一种具有访问控制的云平台下外包数据流动态可验证方法

孙 奕[1),2),3)] 陈性元[2)] 杜学绘[2),3)] 徐 建[2)]

[1)](北京交通大学计算机与信息技术学院 北京 100044)

[2)](解放军信息工程大学 郑州 450004)

[3)](数学工程与先进计算国家重点实验室 郑州 450004)

摘 要 针对云平台下外包数据流不可信、验证范围不可控等问题,该文提出一种具有访问控制的外包数据流动态可验证方法.该方法的核心思想是利用任意的 Hash 函数、双陷门 Hash 函数和 CP-ABE(Cipertext Policy-Attribute Based Encryption)算法构成一种具有访问控制的动态可认证数据结构(AC-MTAT).该可认证结构可以实现对外包数据流的实时增加、更新和细粒度动态验证.此外,该可认证数据结构不仅能够验证外包数据流的完整性,还能够验证数据流序列的正确性.由于在传统默克尔 Hash 树(MHT)中引入双陷门 Hash 函数,AC-MTAT 的构建过程可以分为两个阶段:离线阶段和在线阶段.这样构建 AC-MTAT 所需的主要代价可以放在离线阶段完成,大大提高了 AC-MTAT 在线实时构建效率.该文首先给出 AC-MTAT 方案的形式化定义和具体构建算法;然后,对 AC-MTAT 方案的安全性进行证明,证明其满足正确性、可验证安全性和访问安全性;最后,分析了方案的实现效率,并通过实现一个 AC-MTAT 原型来评估算法的耗费时间,实验结果显示作者的方案对于外包数据流的验证是高效的和可行的,而且与现有方案相比,该方案在实时增长、高效更新、可控验证以及对数据流的适应性等方面更有优势.

关键词 可认证数据结构;安全外包数据流;访问控制;完整性;密文策略基于属性加密算法;云计算

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2017.00337

Dynamic Authenticated Method for Outsourcing Data Stream with Access Control in Cloud

SUN Yi[1),2),3)] CHEN Xing-Yuan[2)] DU Xue-Hui[2),3)] XU Jian[2)]

[1)](*School of Computer and Information Technology, Beijing Jiaotong University, Beijing* 100044)

[2)](*The PLA Information Engineering University, Zhengzhou* 450004)

[3)](*State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou* 450004)

Abstract Aiming at the problem of untrusted outsourcing data stream and uncontrollable authentic scale, this paper proposes a dynamic authenticated method for outsourcing data stream with access control (AC-MTAT). This method is essentially an authenticated data structure that is composed of arbitrary Hash function, double trapdoor Hash function and CP-ABE (cipertext policy-attribute based encryption), which can support data stream with real time adding, updating and verification with fine-grained access control. Besides, the authenticated data structure can not only verify the integrity of the data stream but also the order of the data stream. Due to the introduction of the double trapdoor Hash function on traditional Merkle Hash Tree, we can make the process of constructing the authenticated data structure into two phases: off-line and on-line

收稿日期:2016-03-22;在线出版日期:2016-09-18.本课题得到国家"八六三"高技术研究发展计划项目基金(2012AA012704,2015AA011705)和河南省科技创新人才计划课题(114200510001)资助.**孙 奕**,女,1979 年生,博士研究生,讲师,主要研究方向为云计算、数据安全、安全交换.E-mail:11112072@bjtu.edu.cn.**陈性元**,男,1963 年生,博士,教授,主要研究领域为网络与信息安全.**杜学绘**,女,1968 年生,博士,教授,主要研究领域为多级安全、云安全.**徐 建**,男,1980 年生,硕士,讲师,主要研究方向为数据安全和密码学.

stages. In this way, the main cost to construct the authenticated data structure can be put in the off-line phase, which greatly improves the efficiency of constructing for the authenticated data structure in on-line phase. First of all, this paper put forward the formalized definition and construction of AC-MTAT; Then, the security of the scheme is proved, including correctness, verifiability and access control security. Finally, this paper analyzes the efficiency of the scheme and estimates the calculation time through an experiment on an AC-MTAT model. The experiment also indicates the high efficiency and feasibility in authentication on outsourcing data stream. Comparing to existing scheme, AC-MTAT scheme has absolute superiority in instant adding, effective updating, controllable verification, and data stream applicability.

Keywords authenticated data structures; secure data stream outsourcing; access control; integrality; ciphertext-policy attribute-based encryption algorithm; cloud Computing

1 引 言

随着云计算、大数据、移动网络的快速发展,一方面,许多应用(如 Web 访问分析、环境感知、实时监控和股票交易等)产生大量的数据流.另一方面,终端的计算能力和计算资源有限,无法实时处理或存储这些大规模数据.因此考虑到成本和代价,当大量的数据流产生时,用户(数据拥有者)通常会将这些数据流外包给云平台进行管理和维护,以便其他用户访问和使用.然而云平台的公开性和开放性使得云平台下的数据变得不可信,云平台很可能会向数据使用者提供伪造的数据结果.此外,对于外包数据流来说,随着终端应用数据的不断产生,数据拥有者不断向云平台上传数据,这样云平台下共享的数据一直处于持续增长和动态变化中.如何对远程不可信环境下,连续的、动态变化的外包数据流进行动态更新,实时验证和受控访问就成为解决云平台下数据流实时共享与安全交换的关键所在.

外包数据流连续、实时、海量、非本地存储等特性使得现有验证方案应用在外包数据流时存在以下挑战:(1)验证者无法将全部数据流缓存后再进行验证;(2)传统的完整性验证方法无法高效地验证流序列的正确性;(3)验证者与证明者之间是非交互的;(4)更新数据时需要用户存储以往变更痕迹,否则无法验证数据更新版本的正确性;(5)验证范围不可控,不能实现对验证者身份的细粒度控制,使得被验证数据存在信息泄露等风险.

基于以上分析,本文在对可认证数据结构研究的基础上,引入双陷门 Hash 函数[1-3]和 CP-ABE 算法[4]对典型的可认证数据结构——Merkle Hash Tree[5-6]进行改进,提出一种具有访问控制的可认证数据结构——AC-MTAT,并在此基础上实现了一种新颖的外包数据流动态可验证方法,为解决云平台下数据流实时共享与安全交换提供技术支撑.

2 相关工作

可认证数据结构本质上是一种新型分布式计算模型[7].Tamassia[8]第 1 次正式描述了可认证数据结构模型,模型中包含 1 个客体的结构化集合 S 和 3 个参与者:源、响应者和用户.源持有 S 的原始版本.当在 S 上执行更新时,源产生结构化认证信息,其包含 1 个具有当前版本 S 时间戳的签名.响应者维持 S 的 1 个副本.响应者与源交互获得 S 的更新及相应的结构化认证信息.响应者也与用户交互回答用户对 S 的查询.此外,响应者返回符合的认证信息,该信息包含由源发布的最新结构化认证信息和应答的 1 个证据.用户在 S 上实施查询,但不能直接与源交互,只能与响应者交互.然而,用户相信源但不相信响应者,因此用户验证来自响应者与应答相关的认证信息.被源和响应者用来存储集合 S 的数据结构,以及由不同参与者执行的查询、更新和验证算法,形成可认证数据结构.

目前实现可认证数据结构的实现方式主要有认证跳表[9]、索引 Hash 链表[10]、MHT(Merkle Hash Tree)和 MAC Tree[11]等.其中 MHT 是最早的可认证数据结构.MHT 是一个二进制 Hash 树,叶子结点是各个数据块的 Hash 值,中间结点是所有子结点 Hash 值连接后的 Hash 值.进行完整性检验时只需要对叶子到根结点的一条或若干条路径进行处理,就可以高效地证明一组数据(块)具有完整性,未

被损坏和篡改. MHT 可以仅通过一个签名就能对树中的每个结点值进行认证，又允许中间结点通过计算 Hash 路径的对数来验证树中结点数量，可以实现对指数数量级数据的验证. 这些优势使其成为一种被广为研究的用于验证远程不可信环境下数据完整性的技术. 因此，本文在 MHT 的基础上，构建具有访问控制的云平台下外包数据流动态可验证方案.

目前与本文研究相关的工作主要涉及两个方面：数据持有性证明(Provable Data Possession，PDP)和数据可检索证明(Proof of Retrievability，POR).

PDP 是 Ateniese 等人[12]于 2007 年首先提出的，可以实现对外包数据的高效验证，这是第一个不分块且支持公开验证的方案. POR 是由 Shacham 等人[13]在 2008 年提出的，是对 PDP 的扩展，不仅能够验证远程数据是否被篡改，还可以利用纠删码技术保证数据的可恢复性. 这两种验证模式的主要思想是利用短签名算法(BLS)或 RSA 签名算法构建同态线性验证标签(homomorphic linear authenticator)，使得用户在不取回数据的情况下，对远程服务器(如云)中数据的完整性进行验证，可以分为确定性验证和概率性验证两类.

为了在有效地验证数据完整性的情况下，同时能够支持数据的动态操作，在后续的研究方案中引入了可认证数据结构. Ateniese 等人[14]提出了一种支持数据动态存储的 SPDP 方案. 该方案可以支持数据的更新和删除，但不支持数据的添加. 由于该方案是基于对称密钥的，所以不能支持公开的验证并且由于每次验证都需要消耗一个密钥，其能够提供的数据验证的次数有限. 2009 年，Erway 等人[15]第 1 次提出了一种能够支持全动态操作的可证明的数据持有方案. 该方案为了解决数据块更新后造成的索引信息的变更问题，在标签计算中删除了 Ateniese 所提出的 PDP 模型中的索引信息，利用基于等级的认证跳表构建数据块的标签信息. 不足之处是执行效率不高.

为了同时支持公开审计和全动态数据操作，Wang 等人[16]引入 MHT，通过对 MHT 树型结构的变换，改变现有验证证据的存储模型，来实现全动态数据操作，并利用 BLS 算法构建数据块的同态认证标签，实现公开审计. Hao 等人[17]和 Yang 等人[18]也对可公开验证下的、可支持动态数据的完整性验证方案进行了讨论. 为了进一步提高验证时的查询效率，Li 等人[19]将 MHT 概念融合到 B+树结构中，提出一种 MB-Tree 树型结构对存储数据进行组织和管理，并通过树型结构的变化来体现对数据的动态操作. Zhu 等人[20-21]采用索引 Hash 链表来支持动态数据的有效更新，并利用分离结构(fragment structure)的方法来减小可公开验证方案中所需的存储验证信息的开销. 为了解决多用户下的云端共享数据问题，Tate 等人[22]提出一种可以支持多用户的，支持动态数据更新的数据完整性验证方案. 但该方案需要基于可信硬件(trusted hardware)来实现. 为了支持对云平台下大数据的细粒度更新验证要求，Liu 等人[23]提出一种具有标记的默克哈希树(RMHT)方案. 该方案基于 BLS 算法为每一个分块数据计算同态线性验证标签. 与之前需要均分数据块大小的方案不同，该方案中数据块大小可以是任意大小的，在 MHT 叶子结点中除了保存数据分块信息以外，增加了标记信息，该标记信息用于记录对每个数据分块再次细分后的数据块个数信息及位置信息，从而利用该标记实现对每个数据块中数据的细粒度更新，而无需更新整个数据块信息. Liu 等人[24]、Feng 等人[25]和 Tan 等人[26]也对现有的云平台下外包数据完整性验证技术进行了总结与展望，但均未涉及对外包数据流的完整性验证问题.

综上所述，以上方案都针对静态文件的动态操作，不适用于对数据流的动态操作. 这些方案的一个共同点是需要对文件进行分块，然后将这些数据块作为 MHT 的叶子结点来构建 Hash 认证树，不同方案中根据其所实现的功能不同，叶子结点存储的附加信息不同. 在动态更新过程中，通过变换 MHT 树型结构(如更新某些结点、合并某些结点、分裂某些结点)来实现对数据的动态操作. 以上构建方法，需要用户在构建 MHT 时，提前确定所有叶子结点的信息，才能生成 MHT. 然而，对于数据流来说，不可能将整个数据流缓存下来，分块后再构建 MHT. 因此，构建面向数据流的 Hash 认证树时，不可能提前确定所有叶子结点信息. 以上方案不适用于对数据流的动态操作和实时验证.

最近，Schroeder 等人[27-28]从另一个角度对 MHT 进行改进，引入变色龙 Hash 函数[29]构造一个变色龙认证树(CAT)，来实现对数据流的动态操作和实时验证，并基于 CAT 构造了一个可验证数据流方案(VDS). 该方案优势是基于 CAT 无需提

前固定叶子结点,对于掌握陷门信息的用户,无需预先计算或重新计算其他叶子结点就能对新元素进行添加和认证,而且在数据流协议的设置阶段中数据可以是未知的.

本文在 CAT 的基础上进行改进,构建一种具有访问控制的动态可认证数据结构(AC-MTAT).与 CAT 不同,在 CAT 中为了防止攻击,陷门碰撞只能使用一次,因此当更新数据时需要重新构建整棵树和验证公钥;本文采用双陷门 Hash 函数,可以计算多次碰撞,无需更新根结点的值和公钥,提高了数据的更新效率,而且通过分期偿还机制将认证树的构建分为离线阶段和在线阶段,大大提高数据增长时可认证数据结构的构建效率,从而提高对实时数据的处理效率.此外,本文提出的方案具有基于属性的细粒度授权验证功能,既不是基于对称密钥的认证,也不是对所有用户的公开验证,只有满足一定访问策略的用户才能验证.这样在云平台下面向多个用户、多个可认证数据结构时,可以实现基于属性的细粒度授权验证,从而能够较好的控制共享平台下数据的可验证范围,保护不同数据用户的隐私,同时在授权用户和验证范围更新时,降低密钥管理的复杂度.

3 具有访问控制的动态可认证数据结构

本节提出一种具有访问控制的动态可认证数据结构,并给出其总体描述及形式化定义.

3.1 总体描述

具有访问控制的动态可认证数据结构本质上是由任意的 Hash 函数、双陷门 Hash 函数和 CP-ABE 算法所组成的一种具有访问控制的双陷门 Hash 认证树,该认证树可以实现一种新颖的认证技术.与传统的 MHT 相比,AC-MTAT 在 Hash 函数的基础上,引入了双陷门 Hash 函数(利用双陷门 Hash 可以实现无密钥泄露的陷门 Hash 函数)和 CP-ABE 算法共同构成一棵二进制树.图 1 展示了一个 AC-MTAT 例子:我们用 D 表示树的深度,树的每一层的高度用 $h=0,\cdots,D-1$ 来表示,则树中叶子结点的高度为 $h=0$,根结点的高度为 $h=D-1$.每个非叶子结点最多有两个孩子结点,树的右结点是通过双陷门 Hash 函数计算的值,左结点由 Hash 函数计算的值;中间结点是两个孩子结点 Hash 值连接后的 Hash 值,根结点是 CP-ABE 对根值进行加密的值,记作 γ,只有满足访问策略的用户才能获取根结点的值.

AC-MTAT 的主要构建思路如下:我们用 H 表示 Hash 函数,用 MTH 表示双陷门 Hash 函数.首先随机选择两个叶子结点的值 l_0 和 l_1,来构建初始化认证树,如图 2 所示,计算 $H(l_0\|l_1)=N_{1,0}$ 作为结点 $N_{1,0}$ 的值,随机选择信息 m 和碰撞值 r,利用双陷门 Hash 函数计算 $MTH(m,r)=N_{1,1}$ 作为结点 $N_{1,0}$ 的兄弟结点即 $N_{1,1}$ 的值.接着,计算 $H(N_{1,0}\|N_{1,0})=N_{2,0}$ 作为其父结点的值,依次类推计算到 $D-1$ 层.然后,利用双陷门 Hash 函数计算 $MTH(m,r_\rho)=\rho$,并利用 CP-ABE 算法对 ρ 进行加密得到 γ.

随着数据的增长,可以实现对初始化认证树结点的动态增加,增加的结点分为两种情况,如果增加的是右结点的叶子结点,则只需要更新陷门碰撞值,如:增加叶子结点 l_2 和 l_3,记 $l_2\|l_3=m'$,利用求碰撞值算法获得 r',使得 $MTH(m,r)=MTH(m',r')$,更新 $N_{1,1}$ 的存储值为 (m',r').如果增加的是左节点的叶子结点,则计算叶子结点的 Hash 值,如果兄弟结点不存在,随机选择信息对 (m_i,r_i) 计算陷门 Hash 值作为其兄弟节点,逐层向上计算,直至父节点为陷门 Hash 值,更新其上存储的碰撞值即可.按照以上的方法不断增加新的叶子结点,直至建满树为止.

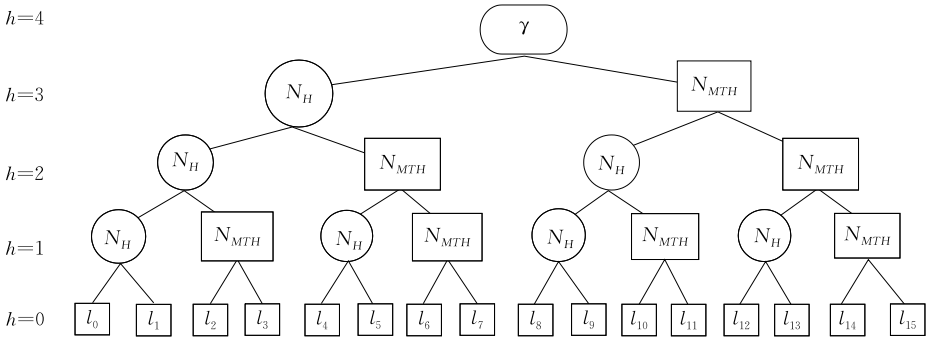


图 1 AC-MTAT 架构

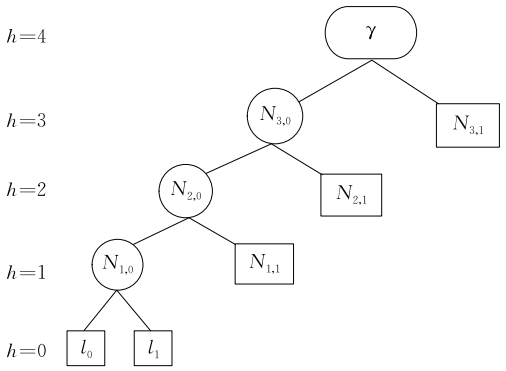


图 2　初始化认证树

3.2　形式化定义

定义 1(双陷门 Hash 函数族 MTH)[2-3]. 一个双陷门 Hash 函数族由一个三元组(I, I', H)所组成：

(1) I 是一种概率多项式时间的密钥生成算法，输入1^k，输出一对长久 Hash/陷门密钥对(HK, TK)，使得 HK, TK 的长度是 k 的多项式有关；

(2) I'是一种概率多项式时间的密钥生成算法，输入 1^k，输出一对一次性 Hash/陷门密钥对(HK', TK')，使得 HK', TK'的长度是 k 的多项式有关；

(3) MTH 是一个随机的 Hash 函数族. 输入1^k，一对 Hash/陷门密钥对(HK, TK)，一对$(m, r) \in M \times R$ 和一个消息 $m \neq m'$，运行求碰撞函数 $MTHCol_{HK}(TK, m, r, m')$输出一个碰撞参数 r'和 HK'，使得 $MTH_{HK}(m, r) = MTH_{HK'}(m', r')$. 当 $HK \neq HK'$时，HK'与其关联的陷门密钥 TK'称为一次性密钥对.

一个双陷门 Hash 函数应满足有效计算、陷门碰撞、抗碰撞、抗密钥泄露和语义安全等性质，详细证明过程参见文献[2-3].

定义 2[4]. 一个基于属性加密的方案由以下 4 个算法组成：$ABE.setup$, $ABE.keygen$, $ABE.encrypt$ 和 $ABE.decrypt$.

$ABE.setup(1^k)$：输入安全参数 1^k，输出一对主公/私钥对(pk, sk).

$ABE.keygen(sk, A)$：主私钥 sk 和属性 A 生成解密密钥 dk_A.

$ABE.encrypt(m, \omega, pk)$：主公钥 pk，消息 m 和访问控制策略 ω，生成密文 c.

$ABE.decrypt(c, dk, \omega)$：如果 A 满足密文 c 绑定的策略 ω，则输入公钥 pk，解密密钥 dk_A 和密文 c，可以解密输出消息 m.

CP-ABE 全安全性证明参见文献[4].

定义 3. 一个具有访问控制的动态可认证数据结构是一种概率多项式时间(PPT)算法 $AC\text{-}MTAT = (MtatGen, IkeyGen, CertIssue, InitiRoot, Add\_Leaf, Update\_Leaf, MtatVrfy)$：

$(SK, VK) \leftarrow MtatGen(1^k, D)$：AC-MTAT 密钥生成算法. 算法输入安全参数 1^k 和一个整数 D，整数 D 表示树的深度. 算法返回一个私钥 SK 和验证密钥 VK，然后发布 VK 并保密 SK.

$(sk_I, pk_I) \leftarrow IkeyGen(1^k)$：IkeyGen 生成算法. 算法输入安全参数 1^k，返回公钥 pk_I 和相应的私钥 sk_I，然后发布 pk_I 并保密 sk.

$cert_A \leftarrow CertIssue(sk_I, pk_I, A)$：属性证书生成算法. 算法输入一对密钥$(sk_I, pk_I)$和一个属性集合 A. 算法输出访问证书 $cert_A$.

$\gamma \leftarrow InitiRoot(pk_I, root, \omega)$：验证树根生成算法. 算法输入公钥 pk_I，认证树根结点 $root$ 和一个访问控制结构 ω. 算法输出关于树根 $root$ 和 ω 的公开验证信息 γ.

$(SK', i, aProof) \leftarrow Add\_Leaf(SK, l)$：认证证据生成算法，算法输入私钥 SK 和一个叶子结点 $l \in L$. 算法输出私钥 SK'，叶子结点 l 在树中的索引 i，以及认证证据 $aProof$.

$(SK', nPath', aPath') \leftarrow Update\_Leaf(SK, VK, i, l')$：叶子结点更新算法. 算法输入一对密钥$(SK, VK)$，叶子结点 $l' \in L$ 及其索引 i. 算法更新位于 i^{th}处的叶子结点为 l'，更新从叶子结点 l'到根路径 $nPath'$上的所有结点，并更新叶子结点 l'认证路径 $aPath'$上的结点.

$\{0,1\} \leftarrow MtatVrfy(VK, i, l, aProof, \gamma)$：验证算法. 算法输入公钥 VK，索引 i，叶子结点 l，认证证据 $aProof$ 和验证根 γ. 如果 l 确实是位于树中 i^{th}的叶子结点，算法输出 1 否则输出 0.

4　AC-MTAT 方案的构建方法

定义 4. 设 H 是一个抗碰撞 Hash 函数，$\widetilde{TH}$ 是单向抗碰撞陷门 Hash 函数，MTH 是一个单向的无密钥泄露的双陷门 Hash 函数，CP-ABE 是一个基于密文策略的属性加密方案. 则具有访问控制的动态可认证数据结构 AC-MTAT 由以下 6 个算法所组成：$AC\text{-}MTAT = (MtatGen, IkeyGen, InitiRoot, Add\_Leaf, Updata\_Leaf, MtatVrfy)$. 下面分别对这 6 种算法进行详细描述.

4.1　初始化参数

$MtatGen$：$MtatGen$ 算法由以下密钥生成算法

组成.

$ThKeyGen(1^k)$:陷门 Hash 密钥生成算法.输入安全参数 1^k,返回一对陷门密钥$(tk,hk)\leftarrow\widetilde{TH}(1^k)$,发布 Hash 密钥 hk,保密陷门密钥 tk.

$MthKeyGen(1^k,D)$:双陷门 Hash 密钥生成算法,输入安全参数 1^k和表示树深度的整数 D.返回长期陷门/Hash 密钥对$(mtk,mhk)\leftarrow MKeyGen(1^k)$和一次性陷门/Hash 密钥对$(\alpha,Y)\leftarrow MKeGen(1^k)$.为了便于描述,用$(\alpha^t,Y^t)$表示当前一次性陷门/Hash 密钥对,则$(\alpha^{t-1},Y^{t-1})$表示上一次更新时的陷门/Hash 密钥对,$(\alpha^{t+1},Y^{t+1})$表示下一次更新时的陷门/Hash 密钥对,当 $t=0$ 时表示双陷门 Hash 函数的长久陷门/Hash 密钥对.

$IkeyGen(1^k)$:输入安全参数 1^k,$IkeyGen$ 算法运行 $ABE.setup$ 算法生成一对密钥$(sk_I,pk_I)\leftarrow ABE.setup(1^k)$,发布 pk_I并保密 sk_I.

$CertIssue(sk_I,pk_I,A)$:对于任意一个属性集合 A,CertIssue 算法运行 $ABE.keygen$ 算法生成解密密钥.然后设 $cert_A:=dk_A$是一个关于 A 的证书.

$InitiRoot(pk_I,root,\omega)$:$InitiRoot$ 算法使用 $Add_Leaf(SK,l)$算法生成树根 ρ,然后输入属性结构 ω 利用算法 $ABE.encrypt(\rho,\omega,pk_I)$生成 γ.因此仅有满足策略 ω 的用户可以利用属性证书解密 γ,并通过 $MtatVrfy$ 算法验证叶子结点.

4.2 构建认证树

$Add_Leaf(SK,l)$路径生成算法在添加叶子结点时分为两个阶段.第 1 阶段:离线阶段,发生在数据流到达之前,第 2 阶段:在线阶段,发生在数据流到达之后.这里设 c 表示计数器,st 表示状态表,h 表示树的高度,并置 $c\leftarrow0$,$st\leftarrow(c,D,[(x,r,Y^t)])$.然后,选择随机数 r,计算$(l_c,l_{c+1})\leftarrow\widetilde{Th}(l_c,r)$.并设 $SK=(tk,mtk,st)$,$VK=(hk,mhk,\gamma)$,发布 VK,保密 SK.

第 1 阶段:离线阶段

(1) $Add_Leaf(off\text{-}line)$算法随机选择叶子结点 l 和数值$r\leftarrow\{0,1\}^k$,计算 l 和 r 的陷门 Hash 函数,将陷门 Hash 值赋予叶子结点 l_0 和 l_1,即$(l_0,l_1)\leftarrow\widetilde{TH}(l,r)$,并计算叶子结点 l_0 和 l_1 的父节点 $n_{1,0}$的 Hash 值即 $n_{1,0}=H(l_0\|l_1)$.随机选择$(x_{h,1},r_{h,1})\in_R M\times R$(for $h=1,\cdots,D-2$),并计算陷门 Hash 值 $n_{h,1}=MTH_{mhk}(x_{h,1},r_{h,1})$,(for $h=1,\cdots,D-2$),随机选择 x_ρ和 r_ρ,计算 $\rho=MTH_{mhk}(x_\rho,r_\rho)$,这里 mhk 表示长期 Hash 密钥,其相应的长期陷门密钥是 msk.计算 $n_{h,0}=H(n_{h-1,0}\|n_{h-1,1})$(for $h=2,\cdots,D-2$)和 $x'_\rho\leftarrow(n_{D-2,0}\|n_{D-2,1})$.最后,计算 $r'_\rho\leftarrow MTHCol(\alpha_\rho^t,x_\rho,r_\rho,x'_\rho,\alpha_\rho^{t+1})$.

(2) 运行算法 $ABE.encrypt$ 生成验证根 $\gamma\leftarrow ABE.encrypt(\rho,w,pk)$.

(3) 生成算法 $MtatVrfy$ 中用于验证叶子结点 l 的认证证据,计算叶子结点 l 的认证证据为 $aPath=((n_{1,1},\cdots,n_{D-2,1}),R,Y)$,并设 $R=(r_\rho,r)$.

(4) 设置计数器 $c\leftarrow2$ 和状态表 $st'\leftarrow(c,D,x'_\rho,r'_\rho,[x,r,Y_i],l_0,l_1)$.算法返回私钥 $SK'=(tk,mtk,st')$,索引序号为 0 和认证证据 $aProof$.

第 2 阶段:在线阶段

$Add_Leaf(on\text{-}line)$算法从状态表中恢复计数器 c 的值.如果 $c>0$ 则运行 $Add\_Leaf(on\text{-}line)$ 算法,添加新的叶子结点及认证证据生成过程如下.

算法 1. $Add\_Leaf(on\text{-}line)$.

for $h=1$ to $D-2$ do
if $\lfloor c/2^h\rfloor$ is even then
$n_{h,\lfloor c/2^h\rfloor}=H(n_{h-1,\lfloor c/2^{h-1}\rfloor},n_{h-1,\lfloor c/2^{h-1}\rfloor+1})$
if $(n_{h,\lfloor c/2^h\rfloor+1})\notin st$ then
$x_{h,\lfloor c/2^h\rfloor+1}\leftarrow\{0,1\}^{2len}$
$r_{h,\lfloor c/2^h\rfloor+1}\leftarrow\{0,1\}^k$
$n_{h,\lfloor c/2^h\rfloor+1}=MTH_{mhk}(x_{h,\lfloor c/2^h\rfloor+1},r_{h,\lfloor c/2^h\rfloor+1})$
$st.add(x_{h,\lfloor c/2^h\rfloor+1},r_{h,\lfloor c/2^h\rfloor+1},Y^t)$
$aPath.add(n_{h,\lfloor c/2^h\rfloor+1})$
else
$aPath.add(n_{h,\lfloor c/2^h\rfloor+1})$
end if
end if
if $\lfloor c/2^h\rfloor$ is odd then
if $(n_{h,\lfloor c/2^h\rfloor})\in st$ then
$x'_{h,\lfloor c/2^h\rfloor}=(n_{h-1,\lfloor c/2^{h-1}\rfloor}\|n_{h-1,\lfloor c/2^{h-1}\rfloor+1})$
$r'_{h,\lfloor c/2^h\rfloor}\leftarrow MTHCol(msk,x_{h,\lfloor c/2^h\rfloor},r_{h,\lfloor c/2^h\rfloor},x'_{h,\lfloor c/2^h\rfloor},\alpha_{h,\lfloor c/2^h\rfloor}{}^{t+1})$
$n_{h,\lfloor c/2^h\rfloor-1}=H(n_{h-1,\lfloor c/2^{h-1}\rfloor-2}\|n_{h-1,\lfloor c/2^{h-1}\rfloor-1})$
$R.add(r'_{h,\lfloor c/2^h\rfloor})$
$Y.add(Y_{h,\lfloor c/2^h\rfloor}{}^{t+1})$
$st.add(x'_{h,\lfloor c/2^h\rfloor},r'_{h,\lfloor c/2^h\rfloor},Y_{h,\lfloor c/2^h\rfloor}{}^{t+1})$
$aPath.add(n_{h,\lfloor c/2^h\rfloor-1})$
$st.del(n_{h-1,\lfloor c/2^{h-1}\rfloor-2},n_{h-1,\lfloor c/2^{h-1}\rfloor-1},x_{h,\lfloor c/2^h\rfloor},r_{h,\lfloor c/2^h\rfloor},Y_{h,\lfloor c/2^h\rfloor}{}^t)$
else
end if
$c\leftarrow c+2$
end for
$R.add(r)$
output$(sp',c,(aPath,R,Y))$

4.3　更新认证树

$Update_Leaf(SK,VK,i,l')$叶子节点更新算法.

(1) 更新叶子结点，令 $l\leftarrow l'$，并计算$(l_i,l_{i+1})\leftarrow \widetilde{Th}(l'_i,r_i)$.

(2) 更新从叶子结点 l'到根路径上的所有结点的算法如下.

算法 2.　更新结点算法.

for $h=1$ to $D-2$ do

if $\lfloor i/2\rfloor$ is even then

$n_{h,\lfloor i/2^h\rfloor}=H(n_{h-1,\lfloor i/2^{h-1}\rfloor}\parallel n_{h-1,\lfloor i/2^{h-1}\rfloor+1})$

$x''_\rho\leftarrow n_{D-2,0}\parallel n_{D-2,1}$

$r''_\rho\leftarrow MTHCol(\alpha'_\rho,x'_\rho,r'_\rho,x''_\rho,\alpha_\rho^{t+1})$

$st.add(x''_\rho,r''_\rho,Y_\rho^{t+1})$

$R.add(r''_\rho)$

$Y.add(Y_\rho^{t+1})$

$st.del(x'_\rho,r'_\rho,Y'_\rho)$

end if

if $\lfloor i/2\rfloor$ is odd then

$x'_{h,\lfloor i/2^h\rfloor}=(n_{h-1,\lfloor i/2^h-1\rfloor}\parallel n_{h-1,\lfloor i/2^h-1\rfloor+1})$

$r'_{h,\lfloor i/2^h\rfloor}\leftarrow MTHCol(\alpha_{h,\lfloor i/2^h\rfloor}{}^t,x_{h,\lfloor i/2^h\rfloor},r_{h,\lfloor i/2^h\rfloor},x'_{h,\lfloor i/2^h\rfloor},\alpha_{h,\lfloor i/2^h\rfloor}{}^{t+1})$

$R.add(r'_{h,\lfloor i/2^h\rfloor})$

$Y.add(Y_{h,\lfloor i/2^h\rfloor}{}^{t+1})$

$st.add(x'_{h,\lfloor i/2^h\rfloor},r'_{h,\lfloor i/2^h\rfloor},Y_{h,\lfloor i/2^h\rfloor}{}^{t+1})$

$st.del(n_{h-1,\lfloor i/2^{h-1}\rfloor},n_{h-1,\lfloor i/2^{h-1}\rfloor+1},x_{h,\lfloor i/2^h\rfloor},r_{h,\lfloor i/2^h\rfloor},Y_{h,\lfloor i/2^h\rfloor}{}^t)$

end if

end for

4.4　实时验证

$MtatVrfy(VK,i,l,aProof,\gamma)$验证算法.

(1) 验证者计算$\rho^*\leftarrow ABE.decrypt(cert_A,pk_I,\gamma)$.

(2) 验证算法计算$(l_i,l_{i+1})\leftarrow\widetilde{Th}(l_i,r)$，并按照以下算法，从$h=2,\cdots,D-2$依次计算叶子结点$n_{h,\lfloor i/2^h\rfloor}$.

If $\lfloor i/2^h\rfloor\equiv 1 \bmod 2$

$x\leftarrow n_{h-1,\lfloor i/2^{h-1}\rfloor}\parallel n_{h-1,\lfloor i/2^{h-1}\rfloor+1}$

$n_{h,\lfloor i/2^h\rfloor}\leftarrow MTH_Y(x;r_{h,\lfloor i/2^h\rfloor})$, with $r_{h,\lfloor i/2^h\rfloor}\in R$

If $\lfloor i/2^h\rfloor\equiv 0 \bmod 2$

$x\leftarrow n_{h-1,\lfloor i/2^{h-1}\rfloor-2}\parallel n_{h-1,\lfloor i/2^{h-1}\rfloor-1}$

$n_{h,\lfloor i/2^h\rfloor}\leftarrow H(x)$

(3) 验证者查询索引 i 并获得认证证据 $aProof=(aPath,R,Y)$，认证证据获得过程如下.

for $h=1$ to $D-2$ do

if $\lfloor i/2\rfloor$ is even then

for $h=1$ to $D-2$

$aPath=aPath.add(n_{h,\lfloor i/2^h\rfloor+1})$

$R=R.add(R_{h,\lfloor i/2^h\rfloor+1})$

$Y=Y.add(R_{h,\lfloor i/2^h\rfloor+1})$

end if

if $\lfloor i/2\rfloor$ is odd then

for $h=1$ to $D-2$

$aPath=aPath.add(n_{h,\lfloor i/2^h\rfloor-1})$

$R=R.add(R_{h,\lfloor i/2^h\rfloor-1})$

$Y=Y.add(R_{h,\lfloor i/2^h\rfloor-1})$

end if

$aProof=(aPath,R,Y)$

end for

output($aProof$)

(4) 验证者通过$\bar{\rho}\leftarrow MTH_Y(n_{D-2,0}\parallel n_{D-2,1};r_\rho)$ (with $r_\rho\in R$)计算根结点ρ. 如果$\bar{\rho}=\rho^*$则叶子结点可以被认证，否则被拒绝.

5　AC-MTAT方案的安全性和效率分析

5.1　安全性证明

5.1.1　正确性分析

定义 5.　如果对于任意一个可证安全的双陷门 Hash 函数和访问控制策略 ω，IkeyGen 生成算法生成密钥对$(sk_I,vk_I)\leftarrow IkeyGen(1^k)$，相对于任意的属性集合 A，属性证书生成算法生成证书 $cert_A\leftarrow CertIssue(sk_I,pk_I,A)$，验证树根生成算法生成验证根 $\gamma\leftarrow InitiRoot(pk_I,root,\omega)$，使得对于任意的 $l\in L$ 和属性集合 A，如果 $A\models w$，$\rho^*\leftarrow ABE.decrypt(cert_A,pk_I,\gamma)$，$\bar{\rho}\leftarrow MTH_Y(n_{D-2,0}\parallel n_{D-2,1};r_\rho)$，可以输出$\{1\}\leftarrow MtatVrfy(VK,i,l,aProof,\gamma)$，则具有 n 个叶子结点的有效算法 $AC\text{-}MTAT=(MtatGen,IkeyGen,CertIssue,InitiRoot,Add_Leaf,Update_Leaf,MtatVrfy)$是正确的.

当客户端和服务器是可信的，方案的输出应该是$\bar{\rho}\leftarrow MTH_Y(n_{D-2,0}\parallel n_{D-2,1};r_\rho)$. 具体地说，如果用户的属性集合 A 满足访问策略 ω，则它可以通过查询的方式执行 $\rho^*\leftarrow ABE.decrypt(cert_A,pk_w)$ 解密 γ. 然后，比较$\bar{\rho}$和 ρ^*，如果$\bar{\rho}=\rho^*$，则叶子结点可以被认证，否则拒绝. 因此方案的正确性依赖于单向的双陷门 Hash 函数和 CP-ABE 的全安全性.

5.1.2　可验证性分析

AC-MTAT 的安全性归约为未使用 CP-ABE 的双陷门 Hash 认证树(MTAT)的安全性和用于加密树根的 CP-ABE 算法的安全性. 在 AC-MTAT 中，只有符合访问控制策略的用户才能够解析验证根 γ，从中得到 ρ 后才能对结点数据进行验证. 下面我们将 AC-MTAT 的可验证安全性与访问安全性分别

进行说明.在证明可验证安全性时,我们假设攻击者可以攻破 CP-ABE 的安全性,即攻击者可以通过 $\rho \leftarrow Decrypt(\gamma,\omega)$ 获得 ρ.因此,我们定义 AC-MTAT 是可验证安全的,如果 AC-MTAT 满足以下安全性质.

(1)攻击者不能够改变 MTAT 的结构.

(2)攻击者不能够改变树中叶子结点的序列.

(3)攻击者不能替代任意一个叶子结点.

(4)攻击者不能向 MTAT 中增加新叶子结点.

攻击模型.下面通过定义一个挑战者 $\mathcal{C}$ 和攻击者 $\mathcal{A}$ 之间的交互式游戏 $G_A^{\text{AC-MTAT}}(1^k)$ 来形式化描述以上属性,这里 1^k 表示安全参数.本节定义的攻击模型建立在文献[27]定义的 CAT 安全模型的基础之上.挑战者生成一对密钥 (SK,PK) 并将公钥 PK 转发给攻击者 $\mathcal{A}$.攻击者 $\mathcal{A}$ 向挑战者随机发送 z 个叶子结点的值 $l_1,\cdots,l_{z(k)}$,挑战者向攻击者返回叶子结点相应认证证据 $aProof_1,\cdots,aProof_{z(k)}$.然后,攻击者 $\mathcal{A}$ 通过输出一个位于树中某个特定位置的且尚未增加叶子结点处的叶子结点,来破坏 MTAT 的结构.更加形式化的描述如下:

建立阶段:挑战者运行算法 $MtatGen(1^k,D)$ 计算出私钥 SK 和一个验证公钥 VK.攻击者 $\mathcal{A}$ 可以获得公钥 VK.

插入问询:这是一个随机过程,攻击者 $\mathcal{A}$ 向挑战者发送一组叶子结点 $l\in L$ 进行问询.挑战者随机选择一个数值 $r\leftarrow\{0,1\}^k$ 并计算 $x\leftarrow\widetilde{Th}(l,r)$,然后运行算法 $(sk',i,aProof)\leftarrow Add\_Leaf(sk,l)$ 并返回相应的 $(i,aProof)$ 给攻击者 $\mathcal{A}$.这里用 $Q:=\{(l_1,1,aProof_1),\cdots,(l_{z(k)},z(k),aProof_{z(k)})\}$ 表示一串问询-应答序列对.另外将 (l,r) 记录到表 T 中.

更新问询:这是一个随机过程.攻击者 $\mathcal{A}$ 发送消息 $(i,l_n')(1\leq n\leq z)$ 向挑战者进行更新问询.攻击者随机选择数值 $r\leftarrow\{0,1\}^k$,并计算 $x'\leftarrow\widetilde{Th}(l',r)$,然后运行更新算法 $(sk',nPath',aProof')\leftarrow Update\_Leaf(sk,i,x')$ 这里 $aProof'=(aPath',R',Y')$,并向攻击者 $\mathcal{A}$ 返回 $(i,aProof')$.这里用 $Q:=\{(l_1,i,aProof_1),\cdots,(l_{z(k)},i,aProof_{z(k)})\}$ 表示一串问询-应答序列对.

输出阶段:最后,如果攻击者能够输出下面任意一种类型的结果,则称攻击者赢得游戏.

类型-1:攻击者 $\mathcal{A}$ 输出 x_i^*,并且 $x_i^*=x_i$,这里 x_i 表示存储在 MTAT 中 i 处的初始值.

类型-2:攻击者 $\mathcal{A}$ 输出 $(l^*,i^*,aProof^*)$,$1\leq i^*\leq z(k)$,$(l^*,i^*,aProof^*)\notin Q$,并且 $MtatVrfy(vk,i^*,l^*,aProof^*)=1$.

类型-3:攻击者 $\mathcal{A}$ 输出 $(l^*,i^*,aProof^*)$,$z(k)<i^*\leq n$ 并且 $MtatVrfy(vk,i^*,l^*,aProof^*)=1$.

攻击者 $\mathcal{A}$ 赢得上述游戏的概率定义为 $Adv_A^{\text{AC-MTAT}}$.

定义 6. 对于任意的 $z\in N$,具有 n 个叶子结点的有效算法 $AC\text{-}MTAT=(MtatGen,IkeyGen,CertIssue,IniitiRoot,Add\_Leaf,Updata\_Leaf,MtatVrfy)$ 是可验证安全的,若对任意的攻击者 $\mathcal{A}$ 在多项式时间内赢得上述游戏的优势 $Adv_A^{\text{AC-MTAT}}$ 是可忽略的.

定理 1. 如果 H 是抗碰撞 Hash 函数,$\widetilde{TH}$ 是单向抗碰撞陷门 Hash 函数,MTH 是一个满足定义 1 中安全属性的无密钥泄露的双陷门 Hash 函数,则 AC-MTAT 满足可验证性.

证明. 证明过程遵循文献[27]的证明过程.假设这里存在一个攻击者 $\mathcal{A}$ 能够以不可忽略的概率赢得以上定义的游戏,则攻击者 $\mathcal{A}$ 至少能够输出类型-1、类型-2 或类型-3 中的一种结果.

类型-1:如果攻击者 $\mathcal{A}$ 能够输出类型-1 的结果,意味着这里存在一个关于叶子结点 l 的陷门碰撞,也就是说存在 $\widetilde{TH}(l_i^*,r_i^*)=x_i^*=x_i=\widetilde{TH}(l_i,r_i)$.然而这与假设 $\widetilde{TH}$ 是抗碰撞的相矛盾.当 $x_i^*\neq x_i$ 时,如果攻击者 $\mathcal{A}$ 能够输出类型-2 或类型-3 的结果,意味着攻击着能够伪造位于 i^* 的叶子结点 l^* 的认证证据.下面,我们分别对这两种类型的伪造攻击进行详细讨论.

类型-2:当攻击者 $\mathcal{A}$ 输出类型-2 的伪造,也就是说攻击者 $\mathcal{A}$ 能够返回一个元组 $(l^*,i^*,aProof^*)$ 且 $1\leq i^*<z$.这意味者我们可以构建一个算法 $\mathcal{B}$ 要么违背 H 的抗碰撞性,要么违背 MTH 的抗碰撞性.算法 $\mathcal{B}$ 的操作过程如下.

(1)建立阶段

算法 $\mathcal{B}$ 从 $\{0,1\}^k$ 中随机选择 z 个叶子结点 $l_1,\cdots,l_z$ 和 z 个字符串,并保存在表 T 中.算法 $\mathcal{B}$ 运行算法 $(tk,hk)\leftarrow THKeyGen(1^k)$ 生成一对陷门/Hash 密钥并计算输出 $x_i\leftarrow\widetilde{Th}(l_i;r_i)(i=1,\cdots,z)$.然后,算法 $\mathcal{B}$ 随机选择一组哑结点 $n_i\leftarrow\{0,1\}^{len}$ 用于和 z 个叶子结点构造一棵深度为 $D=poly(k)$ 的树并从底层叶子结点到顶部计算出树根值 ρ.此外算法 $\mathcal{B}$ 计算 $((l_1,aProof_1),\cdots,(l_z,aProof_z))$ 认证证据,设置公钥 $VK\leftarrow(th,mhk,\rho)$ 并基于公钥 VK 运行一个攻击者 $\mathcal{A}$ 的黑盒模拟器.

插入问询:每当攻击者 $\mathcal{A}$ 希望将叶子结点 l 添加到树中时.攻击者 $\mathcal{A}$ 随机选择一个新鲜值,计算并

设置 $x \leftarrow \widetilde{Th}(l;r)$，然后算法 $\mathcal{B}$ 计算认证证据 $(SK', i, aProof) \leftarrow addLeaf(sk, x)$，并返回 $aProof = (aPath, i, r)$ 给攻击者 $\mathcal{A}$，并将 (l, r) 记录到表 T 中.

(2) 更新问询

每当攻击者 $\mathcal{A}$ 希望将位于 i^{th} 的叶子结点 l 更新为 l' 时. 算法 $\mathcal{B}$ 依照下面方式应答攻击者 $\mathcal{A}$ 的 (i, l') 更新问询. 首先算法 $\mathcal{B}$ 运行求碰撞函数 $MTHCol$，计算出 MTH 的一个碰撞值 $r' \leftarrow MTHCol(\alpha^t, l, r, l', \alpha^{t+1})$，接着将位于 i^{th} 的叶子结点 l 更新为 l'，r 更新为 r' 并保存一次性 Hash 密钥 Y^{t+1}. 然后，算法 $\mathcal{B}$ 运行认证证据生成算法 $(sk', i, aProof) \leftarrow Add\_Leaf(sk, l')$，更新叶子结点 l' 的认证路径 $aPath'$ 上所有结点，并运行更新算法 $Update\_Leaf(SK, VK, i, l')$，更新位于从叶子结点 l' 到根路径 $nPath'$ 上结点信息. 最后，算法 $\mathcal{B}$ 返回认证证据 $aProof = (aPath, i, r, Y^{t+1})$ 给攻击者 $\mathcal{A}$.

(3) 输出阶段

最后，算法 $\mathcal{B}$ 返回 $(x^*, i^*, aProof^*)$，攻击者输出 $(l^*, i^*, aProof^*) \notin Q$（这里 $aProof^* = (aPath^*, R^*, Y^*)$，$x^* \leftarrow \widetilde{Th}(l^*; r^*)$）后停止运行.

由于攻击者 $\mathcal{A}$ 运行在多项式时间内，所以算法 $\mathcal{B}$ 是有效的. 基于文献[27]的描述，这意味着存在一个 Hash 碰撞 $H(aPath_i \| nPath_i) = H(aPath_i^* \| nPath_i^*)$，或者一个陷门 Hash 碰撞 $MTH(aPath_i nPath_i) = MTH(aPath_i^* \| nPath_i^*)$，这里 $nPath^* = (v^*_{1,\lfloor i/2 \rfloor}, \cdots, V^*_{D-2,\lfloor i/2^{D-2} \rfloor})$. 这与 MTH 或 H 是抗碰撞的相矛盾.

类型-3：当攻击者 $\mathcal{A}$ 输出类型-3 的伪造，也就是说攻击者 $\mathcal{A}$ 能够返回一个元组 $(l^*, i^*, aProof^*)$ 且 $z+1 \leq i^* < 2^D$. 这意味着我们可以构建一个算法 $\mathcal{B}$ 要么破坏 MTH 的抗一次碰撞性，要么破坏 MTH 的单向性.

观察攻击者 $\mathcal{A}$ 输出的响应认证路径，该路径上必然包含了一个这样的结点，该结点是 l_z 结点的认证路径上的结点，且该结点必然是由双陷门 Hash 函数计算出的结点. 而且更新问询并不能改变结点上原有的陷门 Hash 值. 这样我们可以构建一个算法 $\mathcal{B}$ 要么破坏 MTH 的单向性，要么破坏 MTH 的抗碰撞性或无密钥泄露性. 下面描述如何构建算法 $\mathcal{B}$ 的推导过程.

(1) 建立阶段

算法 $\mathcal{B}$ 运行一个黑盒模拟器并返回 z 个叶子结点 $l_1, \cdots, l_z$ 给攻击者 $\mathcal{A}$. 攻击者 $\mathcal{A}$ 选择由双陷门 Hash 函数 $n_j^* \leftarrow MTH_Y(x_j^*; r_j^*)$ 计算出的 t 个陷门 Hash 结点. 然后算法 $\mathcal{B}$ 使用 z 个叶子结点和 t 个陷门 Hash 结点 $n_1, \cdots, n_j^*, \cdots, n_t$ 构造一棵深度为 $D = poly(k)$ 的树并从底层叶子结点到顶部计算出树根值 ρ.

(2) 插入问询

每当攻击者 $\mathcal{A}$ 希望将叶子结点 l 添加到树中时. 算法 $\mathcal{B}$ 从表 T 中恢复一对值 (x_i, r_i)，并计算陷门碰撞值 $r' \leftarrow MTHCol(mtk, l, r, l', Y^t)$. 算法 $\mathcal{B}$ 返回 $aProof = (aPath, i, r)$ 给攻击者 $\mathcal{A}$.

每当攻击者 $\mathcal{A}$ 希望将位于 i^{th} 的叶子结点 l 更新为 l' 时. 算法 $\mathcal{B}$ 依照下面方式应答攻击者 $\mathcal{A}$ 的 (i, l') 更新问询. 首先算法 $\mathcal{B}$ 运行求碰撞函数 Col，计算出 MTH 的一个碰撞值 $r' \leftarrow MTHCol(\alpha^t, l, r, l', \alpha^{t+1})$，接着将位于 i^{th} 的叶子结点 l 更新为 l'，r 更新为 r' 并保存一次性 Hash 密钥 Y^{t+1}. 然后，算法 $\mathcal{B}$ 运行认证证据生成算法 $(sk', i, aProof) \leftarrow Add\_Leaf(sk, l')$，更新叶子结点 l' 的认证路径 $aPath'$ 上所有结点，并运行更新算法更新位于从叶子结点 l' 到根路径 $nPath'$ 上结点信息. 最后，算法 $\mathcal{B}$ 返回认证证据 $aProof = (aPath, i, r, Y^{t+1})$ 给攻击者 $\mathcal{A}$.

(3) 更新问询

当攻击者 $\mathcal{A}$ 向挑战者发送 (i, l') 希望更新 i^{th} 处的叶子结点为 l'. 算法 $\mathcal{B}$ 从表 T 中恢复一组值 (x_i, r_i, Y^t) 并计算无密钥泄露的双陷门 Hash 函数的一个碰撞值 $r' \leftarrow Col(Y^t, l, r, l', Y^{t+1})$. 然后，算法 $\mathcal{B}$ 运行 $(sk', i, aProof) \leftarrow Add\_Leaf(sk, l')$，更新叶子结点 l' 的认证路径 $aPath'$ 上所有结点，并将一次性 Hash 密钥 $Y^t \leftarrow Y^{t+1}$ 保存在表 T 中. 最后将认证证据 $aProof = (aPath, i, r, Y^t)$ 返回给攻击者 $\mathcal{A}$.

(4) 输出阶段

最后攻击者输出 $(l^*, i^*, aProof^*) \notin Q$（这里 $aProof^* = (aPath^*, R^*, Y^*)$，$x^* \leftarrow \widetilde{Th}(l^*; r^*)$），$z+1 \leq i^* < 2^D$ 且 $MtatVrfy(PK, i^*, l^*, aProof^*) = 1$ 后停止运行.

我们观察攻击者 $\mathcal{A}$ 的输出元组 $(l^*, i^*, aProof^*)$，这里 $aProof^* = (aPath^*, i^*, R^*, Y^*)$. 设 $nPath^* = (v^*_{1,\lfloor i/2 \rfloor}, \cdots, v^*_{D-2,\lfloor i/2^{D-2} \rfloor})$ 表示从叶子结点 l^* 到树根的路径. 因为 $z+1 \leq i^* < 2^D$，则结点 $n_1, \cdots, n_j^*, \cdots, n_t$ 中至少存在一个结点位于 l^* 的认证路径上. 由于 t 是一个多项式，因此算法 $\mathcal{B}$ 能够以不可以忽略的概率猜测出树中的索引序列. 假设算法猜测的索引序列是正确的，则这里存在一个索引 j 使得 $n_j' = n_j^*$. 这意味着算法 $\mathcal{B}$ 能够利用 $n_j^* \leftarrow MTH_Y(x_j^*, r_j^*)$ 获得一对值 (x_j^*, r_j^*) 使得 $n_j' \leftarrow MTH_Y(x_j^*, r_j^*)$. 如果 $(x_j^*,$

$r_j^*)=(x_j',r'_j)$，则算法$\mathcal{B}$破坏了 MTH 的单向性，如果$(x_j^*,r_j^*)\neq(x_j',r_j')$，则算法$\mathcal{B}$破坏了 MTH 的抗碰撞性或无密钥泄露性. 更进一步讲，假设攻击者$\mathcal{A}$能够以不可忽略的概率$\varepsilon(k)$赢得游戏，则算法$\mathcal{B}$能够以不可忽略的概率$\varepsilon(k)/t$获得成功. 证毕.

5.1.3 访问安全性分析

在 AC-MTAT 方案的证明中，仅满足访问策略的用户能够检测和验证数据流. 在认证树的构造过程中，用户只有解密γ并获得证据后才能够验证外包数据流. 我们假设从 AC-MTAT 获得的认证证据是正确的，则意味着当且仅当用户获得ρ后，才能够检测和验证外包数据流. 因此通过在挑战者和攻击者$\mathcal{A}$之间建立一个交互式游戏$G_A^{\text{Access}}(1^k)$来形式化以上过程. 形式化描述过程如下：

系统建立：挑战者运行算法$IskeyGen(1^k)$生成私钥sk_I和一个验证密钥pk_I. 将pk_I发送给攻击者$\mathcal{A}$，计算$\gamma\leftarrow Initialization(k,pk_I,\rho,\omega)$.

查询阶段：这是一个随机过程，攻击者$\mathcal{A}$随机发送一系列询问$\omega_1,\omega_2,\cdots,\omega_q$给挑战者，挑战者返回相应认证密钥$cert_{\omega_1},\cdots,cert_{\omega_q}$. 挑战者计算$\rho_i\leftarrow Decrypt(cert_{\omega_i},\gamma)$并将$\rho_i$返回给攻击者$\mathcal{A}$. 将一组有序询问-应答对记作$Ce:=\{(cert_{\omega_1},\rho_1),\cdots,(cert_{\omega_q},\rho_q)\}$.

输出：最后攻击者$\mathcal{A}$输出一对$(cert,\rho)$和$MTAT.Verify(auth,i,l,pk,\rho)=1$后停止，则攻击者赢得游戏.

我们定义攻击者$\mathcal{A}$赢得游戏的概率优势为adv_A^{Access}.

定义 7. 如果对于任意的多项式时间攻击者$\mathcal{A}$赢得游戏的概率adv_A^{Access}是可忽略的，则 AC-MTAT 是访问安全的.

定理 2. 如果 AC-MTAT 方案是可验证安全的并且 CP-ABE 方案是全安全的，则 AC-MTAT 满足访问安全性.

证明. 如果这里存在一个攻击者$\mathcal{A}$能够赢得游戏$G_A^{\text{Access}}(1^k)$，则这里要么存在一个攻击者能够以不可忽略的概率破坏 AC-MTAT 的可验证安全性，要么存在一个攻击者$\mathcal{A}$能够以不可忽略的概率破坏 CP-ABE 的安全性. 定理 1 证明了 AC-MTAT 满足可验证安全性. 基于定理 1 的证明，如果 CP-ABE 是全安全的，则 AC-MTAT 满足访问安全性.

这里存在一个攻击者以不可忽略的概率赢得游戏G_A^{Access}则存在一个算法$\mathcal{B}$能够破坏 CP-ABE 的全安全性.

系统建立：算法$\mathcal{B}$获得 CP-ABE 的公钥pk_I. 则算法$\mathcal{B}$通过运行$Initialization(k,pk_I,\rho,\omega)$和属性集合$\omega$生成$\gamma$，这里$\rho=Decrypt_{ABE}(cert_A,\gamma)$，并发送$(pk_I,\gamma)$给攻击者$\mathcal{A}$.

查询阶段：当攻击者$\mathcal{A}$为属性A_i生成问询，这里A_i不满足ω，算法$\mathcal{B}$使用自己的随机预言机应答$\mathcal{A}$的问询. 算法$\mathcal{B}$选择一个随机值t，并发送(t,ρ)给挑战者. 收到应答 C 后，$\mathcal{B}$设置 C 为应答值.

输出：最后，如果$\mathcal{A}$的输出是ρ和$MTAT.Verify(aProof,i,l,pk,\rho)=1$，则$\mathcal{B}$输出“1”，否则$\mathcal{B}$输出“0”.

由于方案所基于的 CP-ABE 是全安全的，因此这里不存在一个攻击者$\mathcal{A}$能够以不可忽略的概率输出ρ. 对于任意一个攻击者$\mathcal{A}$，我们可以得到$\Pr[G_A^{\text{Access}}=1]\leq negli(k)$，这里$negli()$表示可忽略的函数. 证毕.

5.2 效率分析

本方案的效率主要体现在 Hash 函数、双陷门 Hash 函数以及计算碰撞所需的时间和计算资源. 基于属性的加密 CP-ABE 在方案中以黑盒形式出现，其效率取决于具体方案的使用，因此我们主要以 Hash 函数、无密钥泄露的双陷门 Hash 函数和计算碰撞的计算开销为衡量指标来分析本方案的效率，设$Exp(H)$表示计算 Hash 函数所需时间，$Exp(MTH)$表示计算无密钥泄露的双陷门 Hash 函数所需时间，$Exp(Col)$表示计算碰撞所需时间. 我们的方案可以用任意一个双陷门 Hash 函数来初始化，但是为了便于分析比较，本方案采用一种基于离散对数假设安全的、无密钥泄露的双陷门 Hash 函数[2]对本方案进行实例化. 与 CAT 中 KR 变色龙 Hash 函数一样，双陷门 Hash 函数同样支持 Bellare 等人[30]提出的批验证技术.

首先从理论上分析算法在动态增加、更新和验证时的效率. 具体分析如下：

(1) 数据流动态增加效率分析

由于可以在相同根下对陷门 Hash 函数多次更新也是安全的，不存在密钥泄露的问题，由此，我们可以基于分期偿还机制在离线阶段构建一棵初始化认证树，初始化认证树构建的过程中，包含了构建认证树时最坏的情况，即向空树增加第 1 个结点时所需时间是最长的. 因此，在在线阶段本方案的效率是高于 CAT 方案的. 根据本方案的算法，当增加叶子结点时，分为两种情况. 如果插入左叶子结点，所消耗时间不大于$(D-3)Exp(H)+(D-2)Exp(MTH)+$

$Exp(Col)$；如果插入右叶子结点，所消耗时间不大于 $Exp(MTH)+Exp(Col)$.

(2) 数据流更新效率分析

同理，对于叶子结点的更新也分为两种情况. 如果更新左叶子结点，所消耗时间不大于 $(D-3)Exp(H)+Exp(Col)$；如果更新右叶子结点，所消耗时间为 $Exp(Col)$. CAT 方案在数据更新时需要重新构建认证树和生成公钥，显然代价较高.

(3) 数据流验证效率分析

验证过程中在不考虑访问控制的情况下，本方案与 CAT 方案的验证效率相同. 在最坏情况下，验证者需要计算 D 次陷门 Hash 函数，即所消耗时间为 $D(Exp(MTH))$. 文献[2]中的无密钥泄露的陷门 Hash 函数同样支持 Bellare 等人[30]提出的批验证技术，从而大大降低计算的复杂度.

接着，在以上分析的基础上，我们实现了 AC-MTAT 的一些主要算法来模拟 AC-MTAT 的工作过程. 采用的硬件设备配置是 Intel Core i5、内存 4 GB 1600 MHz DDR3 RAM. 代码采用 JAVA 1.6 版本编写并基于 Java 安全数据包实现密码操作. 我们通过实现双陷门 Hash 函数[2]和 MD5 Hash 函数来实现认证树.

如表 1 所示，我们进行了 4 组实验，每组实验分别对高度为 10 层、20 层、40 层和 80 层的认证树算法进行评估，每组实验的算法执行 500 次取平均值. 第 1 组实验评估初始化认证树所耗费的平均时间，第 2 组实验分别评估在最好情况下和最坏情况下动态添加数据所需要的平均时间，第 3 组实验分别评估在最好情况下和最坏情况下实时更新数据所需要的平均时间，第 4 组实验分别评估在最好情况下和最坏情况下验证数据所需要的平均时间. 评估结果如表 1 所示，说明我们的方案是有效的和可行的.

表 1　4 种实验时间的比较

建树层数(level)	初始化树时间/ms	增加叶子结点时间/ms	更新叶子结点时间/ms	验证叶子结点时间/ms
10	30	7～13	7～21	22～26
20	53	7～21	7～29	31～32
40	101	8～38	7～45	49～53
80	194	8～72	7～86	82～83

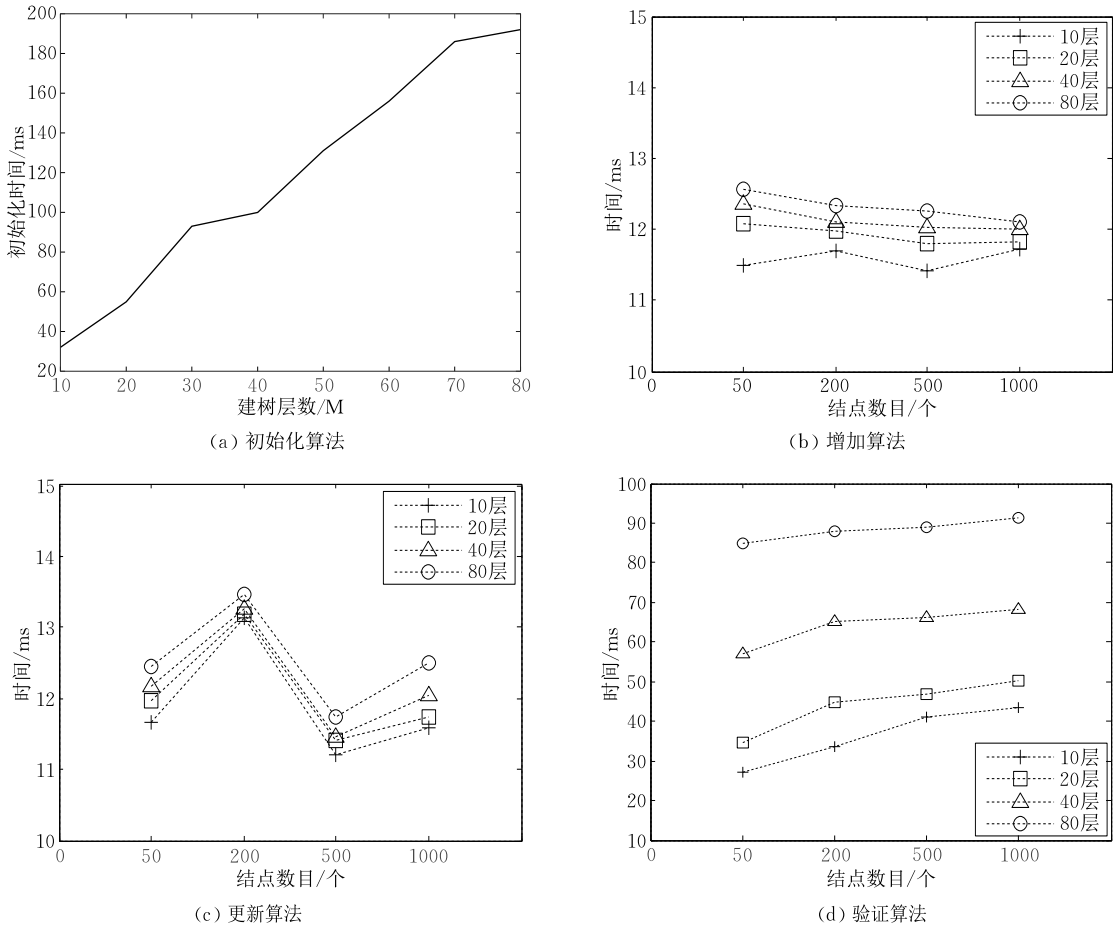
最后，我们对算法在处理大规模结点时的效率问题进一步分析和测试. 分别测试了初始化认证算法、增加算法、更新算法和验证算法在不同建树层次下处理大规模数据的效率.

如图 3(a)所示，在系统初始化阶段，需要构建初始化认证树，其计算开销只与建树层次有关，随着建树层次的增加，初始化认证树的时间随之线性增加. 在新结点加入时，图 3(b)描述了不同建树层次下，新增结点数量对增加算法平均消耗时间的影响，实验数据表明随着建树层次的增加，增加算法的平均消耗时间略微增加，但是在同一层数下，增加结点的规模对增加算法的平均消耗时间影响较小，一直保持相对稳定，说明增加算法具有良好的稳定性. 当更新结点时，图 3(c)描述了建树层数与更新结点数量对更新算法平均消耗时间的影响，随着建树层次的增加，更新结点的平均时间增加量相当微小，表现出对建树层次良好的适应性. 随着更新结点数的增加，更新时间几乎保持不变，表明更新算法在处理大规模数据更新时依然具有良好的性能. 当对结点进行验证时，假设已通过属性证书获得验证根，图 3(d)描述了建树层数与验证结点数量对验证算法平均消耗时间的影响，在验证过程中，建树层次决定了验证路径的长度，随着验证路径长度的增加，相应结点的验证时间也会随之线性增加. 当认证树层数相同时，实验结果表明随着验证结点数量的增加，相应结点的平均验证时间也会线性增加，但增加的趋势越来越小，反应出验证算法在验证大规模数据时的优越性.

5.3　方案比较

与现有典型方案相比，如表 2 所示，本方案在验证者的存储负荷、验证者的计算负荷、服务器的计算复杂度以及通信代价方面的复杂度均为 $O(\log n)$. 此外，在同一共享群中本方案可以实现基于属性的授权验证，既可避免私有验证带来的私钥泄露问题，又能防止公开验证带来的范围过大问题. 通过引入双陷门 Hash 函数，实现了离线初始构建与在线实时更新相结合的动态可验证树，在未获得叶子节点时仍能构建初始可验证树，通过更新碰撞值无需重新计算全部节点信息即可完成可验证树的在线实时更新，在数据持续增长的同时叶子结点动态增加、验证路径动态生成、验证证据实时构建，实现了动态数据边读边写边验证.

本方案支持对大批量数据流的高效动态更新. 大批量数据流传输时必然会存在出错重传问题. 现有方案中一旦出现被检验数据有错误的情形，则数据拥有者必须重传整个数据，而云计算服务提供商必须重新构建整棵验证树. 本方案只需重传出错数据块并更新该数据块从叶子节点到根节点路径上的节点信息，动态更新的代价由 $O(2n-1)$ 降低为 $O(\log n-1)$.



(a) 初始化算法 (b) 增加算法

(c) 更新算法 (d) 验证算法

图 3 平均消耗时间的比较

表 2 方案的特性比较(n 代表叶子节点个数)

方案	CAT [27]	MB-Tree [19]	DPDP [15]	DMHT [16]	本方案
基于属性的授权验证	×	×	×	×	√
流	√	×	×	×	√
实时验证	√	×	×	×	√
离线预处理	×	×	×	×	√
动态更新的代价	$O(2n-1)$	$O(2n-1)$	$O(2n-1)$	$O(2n-1)$	$O(\log n-1)$
验证者的存储负荷	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
验证者的计算负荷	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$
服务器的计算复杂度	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(n)$	$O(\log n)$
通信代价	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$

6 结束语

本文构建了一种新颖的具有访问控制的外包数据流动态可验证方法,设计了离线初始构建与在线实时更新相结合的动态可验证树,分别利用 Hash 函数、双陷门 Hash 函数和 CP-ABE 算法构建可验证树的静态节点、动态节点与根节点,在未获得叶子节点时仍能构建初始可验证树,通过更新碰撞值无需重新计算全部节点信息即可完成可验证树的在线实时更新,在数据持续增长的同时叶子结点动态增加、验证路径动态生成、验证证据实时构建,实现了动态数据边读边写边验证,解决了云平台下外包数据流高效实时验证问题.

参 考 文 献

[1] Ateniese G, De Medeiros B. On the key exposure problem in chameleon Hashes//Proceedings of the 4th International Conference on Security in Communication Networks. Berlin, Heidelgberg, 2005: 165-179

[2] Chandrasekhar S, Chakrabarti S, Singhal M. A trapdoor Hash-based mechanism for stream authentication. IEEE Transactions on Dependable and Secure Computing, 2012, 9(5): 699-713

[3] Sun Yi, Chen Xing-Yuan, Du Xue-Hui. Proxy re-signature scheme for stream exchange. Journal of Software, 2015, 26(1): 129-144(in Chinese)
(孙奕, 陈性元, 杜学绘. 一种用于流交换的代理重签名方案. 软件学报, 2015, 26(1): 129-144)

[4] Lewko A, Okamoto T, Sahai A, et al. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption//Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques. French Riviera, 2010: 62-91

[5] Merkle R C. Protocols for public key cryptosystems//Proceedings of the 1980 IEEE Symposium on Security and Privacy. Oakland, USA, 1980: 122-134

[6] Merkle R C. A certified digital signature//Brassard G ed. Advances in Cryptology—CRYPTO'89, 9th Annual International Cryptology Conference. Santa Barbara, USA, 1990: 218-238

[7] Martel C, Nuckolls G, Devanbu P, et al. A general model for authenticated data structures. Algorithmica, 2004, 39(1): 21-41

[8] Tamassia R. Authenticated data structures//Proceedings of the 11th Annual European Symposium. Budapest, Hungary, 2003: 2-5

[9] Battista G D, Palazzi B. Authentication relational tables and authentication skip lists//Proceedings of the 21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security. Redondo Beach, USA, 2007: 31-46

[10] Zhu Y, Wang H X, Hu Z X. Dynamic audit services for integrity verification of outsourced storages//Proceedings of the ACM Symposium on Applied Computing. New York, USA, 2011: 1550-1557

[11] Yun Y, Shi C, Kim Y. On protecting integrity and confidentiality of cryptographic file system for outsourced storage//Proceedings of the 2009 ACM Workshop on Cloud Computing Security. Chicago, USA, 2009: 67-76

[12] Ateniese G, Johns R B, Curtmola R, et al. Provable data possession at untrusted stores//Proceedings of the 14th ACM Conference on Computer and Communications Security. Alexandria, USA, 2007: 598-609

[13] Shacham H, Waters B. Compact proofs of retrievability//Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security. Melbourne, Australia, 2008: 90-107

[14] Ateniese G, Pietro R D, Mancini L V, Tsudik G. Scalable and efficient provable data possession//Proceedings of the 4th ICST SecureComm. Istanbul, Turkey, 2008: 1-10

[15] Erway C, Küpçü A, Papamanthou C, Tamassia R. Dynamic provable data possession//Proceedings of the 16th ACM Conference on Computer and Communications Security. Chicago, USA, 2009: 213-222

[16] Wang Q, Wang C, Ren K, et al. Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Transactions on Parallel & Distributed Systems, 2010, 22(5): 847-859

[17] Hao Z, Zhong S, Yu N. A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability. IEEE Transactions on Knowledge and Data Engineering, 2011, 23(9): 1432-1437

[18] Yang K, Jia X. 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

[19] Li Ling. Reserch on Some Issues of Data Security in Cloud Computing Services [Ph. D. dissertation]. University of Science and Technology of China, Hefei, 2013(in Chinese)
(李凌. 云计算服务中数据安全的若干问题研究[博士学位论文]. 中国科学技术大学, 合肥, 2013)

[20] Zhu Y, Wang H, Hu Z, et al. Dynamic audit services for integrity Verification of outsourced storage in clouds. IEEE transactions on services computing, 2013, 6(99): 227-238

[21] Zhu Y, Ahn G-J, Hu H, et al. Dynamic audit services for outsourced storage in clouds. IEEE Transactions on Services Computing, 2013, 6(99): 227-238

[22] Tate S R, Vishwanathan R, Everhart L. Multi-user dynamic proofs of data possession using trusted hardware//Proceedings of the ACM Conference on Data and Application Security and Privacy. San, Antonio, Tx, USA, 2013: 353-364

[23] Liu C, Chen J, Yang L T, et al. Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates. IEEE Transactions on Parallel & Distributed Systems, 2014, 25(9): 2234-2244

[24] Liu C, Yang C, Zhang X, et al. External integrity verification for outsourced big data in cloud and IoT: A big picture. Future Generation Computer Systems, 2014, 49(C): 58-67

[25] Feng Chao-Sheng, Qin Zhi-Guang, Yuan Ding. Techniques of secure storage for cloud data. Chinese Journal of Computers, 2015, 38(1): 150-163(in Chinese)
(冯朝胜, 秦志光, 袁丁. 云数据安全存储技术. 计算机学报, 2015, 38(1): 150-163)

[26] Tan Shuang, Jia Yan, Han Wei-Hong. Research and development of provable data integrity in cloud storge. 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)

[27] Schroeder D, Schroeder H. Verifiable data streaming//Proceedings of the 2012 ACM Conference on Computer and Communications Security. Raleigh, USA, 2012: 953-964

[28] Schroeder D, Simkin M. VeriStream—A framework for verifiable data streaming//Proceedings of the 19th International Conference on Financial Cryptography and Data Security. InterContinental San Juan, Puerto Rico, 2015: 548-566

[29] Krawczyk H, Rabin T. Chameleon signatures//Proceedings of the ISOC Network and Distributed System Security Symposium. San Diego, USA, 2000

[30] Bellare M, Garay A, Rabin T. Fast batch verification for modular exponentiation and digital signatures//Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques. Espoo, Finland, 1998: 236-250

SUN Yi, born in 1979, Ph. D. candidate, lecturer. Her current research interests include cloud computing, data security and secure exchange.

CHEN Xing-Yuan, born in 1963, Ph. D., professor. His current research interests include network and information security.

DU Xue-Hui, born in 1968, Ph. D., professor. Her current research interests include multi-level security and cloud security.

XU Jian, born in 1980, M. S., lecturer. His current research interests include data security and cryptography.

Background

Many applications produce large amounts of data stream, such as Web access analysis, sensor network monitoring, real-time monitoring, distributed intrusion detection and stock trading. Considering the cost and expense, when these data stream are generated, they are often outsourced to an intermediary (such as cloud, outsourcing service providers) for managing and storing. However, the intermediary may expose to a public and untrusted environment that has potential leakage of information disclosure, tampering and unauthorized access. With all these security weaknesses, the trust and integrity verification of outsourcing data stream is required.

Integrity verification of outsourcing data stream is the research of provable data possession. The provable data possession in cloud has been extensively researched by many papers. But the existing schemes are directed at the outsourcing data, cannot be used to outsourcing data stream. In addition, the existing schemes are either public verification or private verification, there are some risks that security key exposure or over scale verification.

This paper proposes a dynamic authenticated method for outsourcing data stream with access control. Compared with the traditional MHT, the method has no need to determine the leaf node information in advance, and can realize the dynamic increase and update of the leaf nodes with the growth of data stream. It supports real-time verification of data stream and fine-grained verification based on CP-ABE. Besides, the method can not only verify the integrity of the data stream but also the order of the data stream.

This research is supported by the National High Technology Research and Development Program (863 Program) of China (Nos. 2012AA012704, 2015AA011705), and the Science and Technology Innovation Talents Plan of Henan (No. 114200510001).