计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2016.20150158
53(4):904-920,2016

一种面向云存储的动态授权访问控制机制

王 晶 黄传河 王金海
(武汉大学计算机学院 武汉 430072)
(wjing@whu.edu.cn)

An Access Control Mechanism with Dynamic Privilege for Cloud Storage

Wang Jing, Huang Chuanhe, and Wang Jinhai
(*Computer School*, *Wuhan University*, *Wuhan* 430072)

Abstract Cloud storage is a novel data storage architecture. There are some challenges about data security and manageability in cloud. Cloud needs to provide secure and reliable data access service for users. Because of the variety and volume of the data in cloud, a fine-grained access control mechanism named attribute-based encryption (ABE) has been proposed to ensure data security. In ABE mechanism, data owner describes access privileges of data by access policies and encrypts the data with the policy. User can recover the data if and only if he matches with the policy. Due to various reasons, the access privilege is dynamic and changeable, which increases the difficulty of data management and costs lot of system resource in cloud. Thus, we construct a cloud storage architecture provided by fine-grained ciphertext access control mechanism by use of utilizing ABE which supports efficient, security and manageable data access service. Firstly, we propose a transformation method amongst the common types of access policy, such that the access policy is expressed more generaly. Secondly, we provide three methods to manage access policy: updating privilege, agency privilege and temporary privilege. All of the methods can reduce a lot of computation and communication cost brought by policy updating. Finally, we give the analysis and simulation about our scheme. The results show that our cloud storage architecture is security, efficient and manageable.

Key words cloud storage architecture; data security; dynamic privilege; attribute-based encryption (ABE); access control system

摘 要 云存储是一种新型的数据存储体系结构,云存储中数据的安全性、易管理性等也面临着新的挑战.首先,云存储系统需要为用户提供安全可靠的数据访问服务,并确保云端数据的安全性.为此,研究者们针对云存储中数据结构复杂、数据存储量大等特点提出了属性加密(attribute-based encryption, ABE)方案,为云储存系统提供细粒度的密文访问控制机制.在该机制中,数据所有者使用访问策略表示数据的访问权限并对数据进行加密.但数据的访问权限常会因各种原因发生改变,从而导致云中存储密文的频繁更新,进而影响数据的易管理性.为避免访问权限管理造成大量的计算和通信开销,提出了

收稿日期:2015-02-15;**修回日期**:2015-05-14
基金项目:国家自然科学基金项目(61373040,61173137);高等学校博士学科点专项科研基金项目(20120141110002);湖北省自然科学基金重点项目(2010CDA004)
This work was supported by the National Natural Science Foundation of China (61373040, 61173137), Research Fund for the Doctoral Program of Higher Education of China (20120141110002), and the Natural Science Foundation of Hubei Province of China (2010CDA004).
通信作者:黄传河(huangch@whu.edu.cn)

一种高效、安全、易管理的云存储体系结构:利用 ABE 加密机制实现对密文的访问控制,通过高效的动态授权方法实现访问权限的管理,并提出了不同形式的访问策略之间的转换方法,使得动态授权方法更为通用,不依赖于特定的访问策略形式;针对授权执行者的不同,制定了更新授权、代理授权和临时授权 3 种动态授权形式,使得动态授权更为灵活、快捷;特别地,在该动态授权方法中,授权执行者根据访问策略的更改计算出最小增量集合,并根据该增量集合更新密文以降低密文更新代价.理论分析和实验结果表明,该动态授权方法能减小资源的耗费、优化系统执行效率、提高访问控制机制灵活性.

关键词　云存储体系结构;数据安全;动态授权;属性加密;访问控制系统

中图法分类号　TP303; TP309

云存储是一种新型的数据存储体系结构,与传统的存储体系相比,云存储不是单一的硬件设备,而是由网络设备、存储设备、服务器、应用软件、公共访问接口和客户端程序等多个部分组成的复杂系统.云存储以存储设备为核心,通过网络对分布式存储资源进行整合利用,对外提供方便数据存储和访问服务,是以存储为核心为用户提供各种数据服务的分布开放式服务系统.

数据安全问题是云存储体系中至关重要的问题.为确保云中的数据安全,需要为云存储提供一种灵活可靠的数据访问控制机制.显然,传统的粗粒度访问控制机制在分布开放式的云环境中不再适用.为此,Goyal 等人[1]提出了一种细粒度的密文访问控制机制——基于属性加密(attribute-based encryption, ABE)的访问控制机制.ABE 以属性定义密钥,并结合访问策略(access policy)对数据进行加密,从而定义密文的访问权限.访问策略作为 ABE 机制的核心部分,其表示形式较为多样.目前,访问策略主要有 3 种表示形式:单调的布尔公式(monotonic Boolean formulas)、访问控制树[1](access tree)和线性秘密共享矩阵[2](linear secret sharing scheme matrix, LSSS-matrix).此 3 种表示形式各具优点,其中 LSSS 矩阵由于其灵活的表现力及高效的计算方法而被广泛应用于 ABE 的研究领域中.

访问策略是 ABE 实现数据细粒度访问控制的关键,通过对其优化可实现对数据访问权限的高效管理,提高数据的易管理性.事实上,越复杂的访问策略其表达能力越强,所能实现的权限管理也越灵活.然而,复杂的访问策略也增大了系统的计算量和通信量,致使访问策略的管理也需要花费极大的计算和通信代价.在现有的 ABE 密文访问控制机制中,数据所有者在更改某个数据块的访问权限时,需要重新制定访问策略对数据进行重加密并将新生成的密文重新上传至云服务器,这使得数据所有者需要承担较大的计算和通信开销.由此可见,访问策略的变动越频繁,对数据管理造成的负担也越大.因此,为了增加云存储系统中访问控制机制的灵活性,并提高数据权限管理的效率,本文在 ABE 密文访问控制机制中引入动态授权方法,以优化系统的访问权限管理机制,减小因访问策略变更所带来的各方面开销,提高系统的运行效率并增强系统的易管理性.根据云存储体系中不同的应用需求,本文定义 3 类动态授权方式:

1) 更新授权.数据所有者直接对数据的访问策略进行更改.数据加密后被存储在云端,此后,数据所有者若要更改其访问策略,仅需对云端的加密数据进行局部更新即可.

2) 代理授权.数据所有者委托第三方(代理方)对指定数据的部分访问权限进行管理,以实现间接授权.由于某些数据的访权限会频繁变动,而数据所有者无法及时对访问策略进行直接调整,此时,通过将访问策略的部分管理权限委托给第三方代为处理,以实现间接快速授权.

3) 临时授权.授权机构(authority)授予用户的一次性访问权限.紧急情况下,用户需要访问其权限范围外的数据,而无法立即从数据所有者处获取权限,则可从授权机构处申请一次性的临时访问权限.

由于现有 ABE 中访问策略的表示形式多样,为使动态授权方法的应用更为广泛、不受访问策略的形式所限制,该方法不能依赖于某种具体的访问策略而存在,需要具备通用性.同时,计算代价和通信代价是动态授权方法 2 个重要的评测指标,过大的计算代价会导致系统运行效率降低而过大的通信代价则会造成网络拥塞,动态授权方法必须以较低的代价实现对数据访问权限的管理.而现有的大部分文献都只针对特定的访问策略类型而展开研究,且在计算效率及通信效率上并未加以优化.如:文献[1]提出了访问树更新机制;文献[3]提出了 LSSS 矩阵

更新机制. 文献[1,3]所实现的访问策略更新机制都只能在原有的策略上增加新的限制条件;文献[4]提出了更为灵活的访问策略更新机制,能使新的访问策略不受原有策略的限制且对常用的访问策略分别提出了具体的更新方法,但没解决策略结构的高效更改问题,其通用性也存在一定的局限. 此外,文献[4]在处理门限结构时效果并不理想,需要将其转换为与门和或门的组合形式,再进行相应处理,这使得处理过程更为复杂,计算参量更多,降低了执行效率,增大了计算复杂度.

ABE 可以看作是上层的秘密分享方案(secret sharing scheme, SSS)和底层的公钥(public key, *PK*)加密方案的组合. 在本文所提出的 ABE 方案中,上层 SSS 方案与下层公钥加密方案之间相对较为独立,更改上层不会对下层造成影响,反之亦然. 不论是上层的访问策略结构更改还是底层的属性更改,在本文所提出的方案中都能以最快的速度和最小的更新代价实现. 同时,在通用性上,本文在文献[5]的基础上提出由单调布尔公式及访问控制树向 LSSS 矩阵转换的算法,通过该算法所转换的访问策略在逻辑结构及数值结构上都保持不变. 特别地,本中将 LSSS 矩阵根据对应访问策略的结构进行分块,并以"块"为单位实现对访问策略的管理,可灵活地更改访问策略中的任意"块",也可对门限结构直接进行处理,简化了处理流程,加快了处理效率,并能以最小的计算代价及通信代价实现密文更新,从而进一步提高系统的运行效率、灵活性和易管理性.

1 相关研究

云存储的出现为存储系统带来许多新的发展,许多企业建立了面向广大用户的云存储服务系统,如亚马逊的简单存储服务(S3)、RackSpace 的 Openstack 以及微软的 Azure. 同时,云存储也受到的广泛研究者的关注,文献[2,6-8]中均针对云存储的特征提出了适用于网络环境的存储体系结构.

数据安全问题是云存储体系中 1 个至关重要的问题[9-11],访问控制机制是实现云端数据安全的一种有效手段. ABE 是一种支持一对多的公钥加密机制[12],为云存储系统提供了一种细粒度的密文访问控制机制,它根据属性定义密钥并通过访问策略定义数据访问权限,适用于云中复杂的数据环境. 最早提出的 ABE 也被称为模糊的身份加密(fuzzy identity-based encryption, FIBE),由 Sahai 等人于文献[13]中提出. 此后,文献[1,14]在 FIBE 的基础上分别提出了密钥策略属性加密(key-policy attribute-based encryption, KP-ABE)和密文策略属性加密(ciphertext-policy attribute-based encryption, CP-ABE),令数据的访问控制机制更为灵活、更富表现力. 文献[15-17]中均以访问树为基础实现数据细粒度的密文访问控制方案. 此后,文献[18-22]中又根据线性秘密分享方案(linear secret sharing scheme, LSSS)实现了以 LSSS 矩阵为访问策略的 ABE 方案. LSSS 矩阵相对访问树而言计算效率更高,实现更为方便;而文献[4]中利用范德蒙行列式在 LSSS 矩阵中实现了门限结构,增强了 LSSS 矩阵的表现力并提升了加解密运行的效率.

ABE 机制可实现细粒度的访问控制,但其计算及通信开销都较大. 出于对系统性能的考虑,文献[23-25]中提出了对 ABE 机制的优化方案,提高了系统的执行效率;文献[26]中提出的在线/离线方案,通过离线计算的方式降低系统的通信代价. 但是,在访问权限变动频繁的系统中,上述方案都没有提供有效的访问权限管理接口,对系统整体性能的优化有限. 访问权限的管理是基于 ABE 的访问控制系统所要面临的瓶颈问题,也会对系统的整体性能产生极大的影响. 文献[1]中为 KP-ABE 方案提供了最初的策略更新算法用于更新用户密钥,但对权限的管理能力有限,仅能执行少量的管理操作. 文献[3]中提出的 KP-ABE 方案比文献[1]中提出的方案更为完善,但仍只支持向访问策略中动态添加新的约束条件. 文献[4]中提出了支持动态策略更新的 CP-ABE 方案,该方案中针对各种类型的访问策略提出相应的策略更新算法. 但文献[4]中对门限机构的处理需先将其转换为一个合取范式再进行处理,这会增大密文的冗余也会影响加解密等操作执行的效率.

2 问题描述

2.1 问题定义与设计目标

云存储系统是 1 个多设备、多应用、多服务协同工作的集合体,它将大量不同类型的存储设备通过软件技术集合到一起协同工作,共同为数据存储提供服务. 随着云存储的应用越来越广泛,对云存储的安全也提出了更高的要求. 用户在希望能更多地使用云存储服务的同时,也需要云存储服务提供商给予他们足够的安全保证及便捷的管理方法. 云存储

系统在不安全的网络环境中运行,数据通过网络进行传输并存储,网络的安全问题势必会对数据安全性带来影响,造成数据的泄露.同时,云存储中的海量数据存储规模,也会为数据的管理带来极大的困扰,耗费大量的系统资源.因此,数据的安全性及易管理是本文云存储体系设计的重点:

1) 安全性.确保数据在存储和传输过程中的保密性,防止数据泄露.

2) 易管理性.由数据所有者自主管理数据访问权限,为其提供高效可行的权限管理接口.

本文通过 ABE 实现云存储中细粒度的访问控制机制,为用户提供安全可靠的数据访问及存储服务,构建安全的云存储服务体系.该体系中,通过访问策略实现安全可靠的数据访问控制,但访问策略的灵活多变也使得数据访问权限的管理变得复杂且低效.为了实现高效快速的数据访问权限管理,从而提高整个云存储体系的运行效率并增加访问控制机制的灵活性及易管理性,本文为该体系设计了一种高效安全的动态授权方法.该动态授权方法可转化为 2 个函数的定义及求解:

$$\mathcal{F}_{\mathrm{Attr}}:\mathcal{A}\times\mathcal{A}\to\Delta, \tag{1}$$

$$F:CT\times\Delta\to CT. \tag{2}$$

其中,$\mathcal{F}_{\mathrm{Attr}}$ 表示计算 2 个访问策略间的差量集合的函数;$\mathcal{A}$ 表示访问策略空间;Δ 表示差量集合空间;F 表示根据原密文和差量集合更新访问策略生成新密文的函数;CT 表示密文空间.差量集合 Δ 的大小决定了执行策略更新所需花费的计算及通信代价.为实现高效安全的权限管理,函数 $\mathcal{F}_{\mathrm{Attr}}$ 及 F 的设计需要着重考虑 2 个方面:

1) 执行效率.计算出最小的增量集合 Δ,降低策略管理所需的计算及通信代价.

2) 数据安全性.策略管理过程中,要确保数据的前向安全及后向安全.

2.2　相关定义与假设

2.2.1　双线性映射

现有的 ABE 机制大多是基于双线性映射实现的,现给出双线性映射的定义及其复杂性假设如下:

定义 1. 双线性映射(bilinear map).设 G_0,G_1 是 2 个阶为素数 p 的乘法循环群,g 为群 G_0 的 1 个生成元.$e:G_0\times G_0\to G_1$ 为双线性映射,若 e 满足

1) 双线性.$\forall u,v\in G_0$,$\forall a,b\in\mathbb{Z}_p$,有 $e(u^a,v^b)=e(u,v)^{ab}$.

2) 非退化.$e(g,g)\neq 1$.

3) 可计算.$\forall u,v\in G_0$,存在一个有效的多项式时间算法可计算 $e(u,v)$.

假设 1. q-parallel BDHE[17](decisional q-parallel bilinear Diffie-Hellman exponent assumption).设有阶为素数 p 的乘法循环群 G_0,G_1 及双线性映射 $e:G_0\times G_0\to G_1$.现选取随机数 $a,s,b_1,b_2,\cdots,b_q\in\mathbb{Z}_p$ 及 G_0 的生成元 g.记 $\eta_0=e(g,g)^{a^{q+1}s}$,η_1 为 G_1 中任意元素.给定攻击者如下数据:

$$g,g^s,g^a,g^{a^2},\cdots,g^{a^q},g^{a^{q+2}},g^{a^{q+3}},\cdots,g^{a^{2q}},$$

$$\forall j\in\mathbb{Z}_q^*,g^{sb_j},g^{a/b_j},g^{a^2/b_j},\cdots,g^{a^q/b_j},$$

$$g^{a^{q+2}/b_j},g^{a^{q+3}/b_j},\cdots,g^{a^{2q}/b_j},$$

$$\forall j,k\in\mathbb{Z}_q^*,j\neq q,g^{sb_j},g^{asb_j/b_k},g^{a^2sb_j/b_k},\cdots,g^{a^qsb_j/b_k}.$$

在多项式时间内,攻击者无法区分 η_0 和 η_1.即不存在多项式时间敌手在解决上述问题时存在不可忽略的优势.

2.2.2　访问结构

访问结构是访问策略的具体表现形式,其实现方式有多种.目前大多数 ABE 的研究中所定义的访问结构均为单调的,以下给出单调访问结构的通用定义.

定义 2. 单调访问结构[20](monotonic access structure).设 $P=\{P_1,P_2,\cdots,P_N\}$ 为一个属性集合.访问结构 $\mathcal{A}$ 即为幂集 $2^{\{P_1,P_2,\cdots,P_N\}}$ 的一个非空的单调子集.设 P' 为 $P=\{P_1,P_2,\cdots,P_N\}$ 的一个子集,若 $P'\in\mathcal{A}$ 则称为 $\mathcal{A}$ 的可满足集,否则称为 $\mathcal{A}$ 的不可满足集.$\mathcal{A}\subset 2^{\{P_1,P_2,\cdots,P_N\}}$ 是单调的,若 $\forall P',P''$ 有 $P'\in\mathcal{A},P''\supset P'\Rightarrow P''\in\mathcal{A}$.

2.2.3　安全性模型

本方案的安全模型被定义为攻击者 A 及挑战者 B 之间的选择性游戏(selectivity-game),模型具体定义如下:

初始化.攻击者 A 生成 1 个挑战策略 T 并将 T 提交给挑战者 B.

建立.挑战者生成系统公钥(PK)及主钥(master key,MK)并公布给攻击者 A.

阶段 1.攻击者 A 向挑战者 B 发出私钥(secret key,SK)查询,所查询私钥都不能满足访问策略 T.

挑战.A 提交 2 个等长的密文 μ_0 和 μ_1,挑战者 B 随机选取 $b\in\{0,1\}$,并根据访问策略及系统公钥加密 μ_b,并将生成的密文返回给 A.

阶段 2.重复阶段 1.

猜测.攻击者 A 给出对 b 的猜测 b'.

将攻击者 A 在上述游戏中的优势定义如下:

$$Adv_A=\left|Pr(b'=b)-\frac{1}{2}\right|. \tag{3}$$

3 安全易管理的云存储体系

云存储为用户提供方便的数据访问接口的同时也引发了用户对数据安全性的担忧,安全技术问题是云存储技术普及所必须克服的瓶颈问题.基于ABE的密文访问控制系统能为云存储体系提供高效灵活的密文访问控制,是解决云存储中数据安全的关键技术之一.然而,ABE中灵活多变的访问策略为数据的管理带来了极大的困难:访问权限的管理更新必然导致计算和通信开销的急剧增大.因此,需要为基于ABE的密文访问控制机制设计高效便捷的权限管理方法,以此提高整个云存储体系中数据的易管理性.

基于ABE的密文访问控制机制是上层的秘密分享方案及下层的公钥加密机制的组合,所谓的访问策略即由上层的秘密分享方案实现.秘密分享方案可将一个秘密(secret)s根据指定的访问结构进行分解并生成一个分量(shares)集合$S=\{s_1,s_2,\cdots,s_K\}$,每个部分s_i,$1\leqslant i\leqslant K$由1个指定的属性公钥对其进行加密.对应属性私钥可还原相应的s_i,当解密者所拥有的属性私钥集合对应访问结构的1个可满足集时则可还原出s.本文借鉴文献[5]中的矩阵生成算法实现访问策略形式的转换,并构建具有分块特性的矩阵来表示访问策略.由于分块矩阵中各块之间相对较为独立,改动其中1个"块"不会对其他"块"造成影响,因而可用较小的代价实现访问结构的更改,极大地提高访问策略的可扩展性,进而提高数据的易管理性.

3.1 云存储体系结构设计

在本文所设计的云存储系统中,存在4类实体:授权机构、数据所有者、云及用户,如图1所示.各类实体功能如下:

1) 授权机构.授权机构负责生成并管理密钥(包括MK,PK,SK).为便于系统的扩展及功能模块的划分,该系统中设立了多个授权机构:一个中央授权机构(certification authority, CA)和多个属性授权机构(attribute authority, AA).中央授权机构具备最高管理权限,负责生成并管理全局密钥(global key),而属性授权机构负责生成并管理属性密钥(attribute key).

2) 数据所有者.数据所有者为数据制定访问策略并用PK为数据加密,生成密文.

3) 云.云端负责存储数据所有者上传的密文并在用户需要访问数据时将该密文发送给用户.

4) 用户.用户从授权机构处获取与自身属性相关的私钥SK,并通过私钥SK解密具备访问权限的密文.

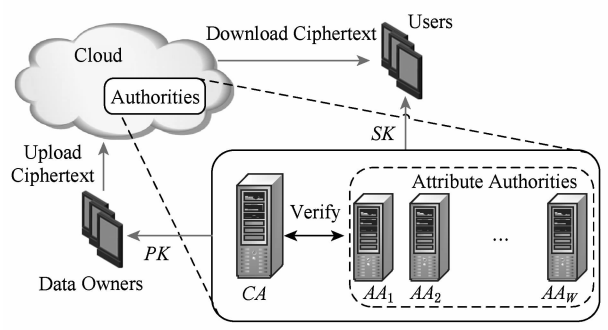


Fig. 1 Architecture.

图1 系统架构

3.2 数据访问控制及权限管理

在上述云存储体系中,CA为全局唯一可信机构,可对用户及AA进行管理;数据所有者可根据需要自主定义访问策略,直接对数据访问权限进行管理;所有数据以密文形式存储于云端,通过加密算法确保云端数据的安全;用户从云端获取密文,并用自身私钥还原出明文.在整个系统中,仅有数据所有者和满足访问策略的用户可获取有效数据,有效解决了云存储体系中的数据安全性问题.

为增强云存储体系中数据的易管理性,本文将根据执行者的不同提出3种动态授权方法:数据所有者执行的更新授权、第三方代理服务器执行的代理授权及CA所执行的临时授权,如图2所示.

1) 更新授权.数据所有者要更改某个数据的访问权限时,先根据该数据访问权限的变动计算最小增量集合,再将该集合上传至云端.云服务器只需根据增量集合对相应密文进行局部更新,即可实现数据访问权限的更改.

2) 代理授权.数据所有者可将数据的部分访问权限委托给第三方(代理服务器)进行管理.数据所有者在加密数据的同时生成1个代理集合,并将该代理集合交付给指定的代理授权服务器.代理服务器可根据代理集合计算更改访问权限所需的增量集合,以实现对数据访问权限的间接管理.

3) 临时授权.在紧急情况下,用户可向CA申请某些数据的临时访问权限.当接收CA到1组用户的临时访问申请时,先根据原密文生成临时密文,再为每个满足条件的用户生成临时会话密钥,最后将临时密文及会话密钥返回给该组用户,令用户获取一次性的临时访问权限.

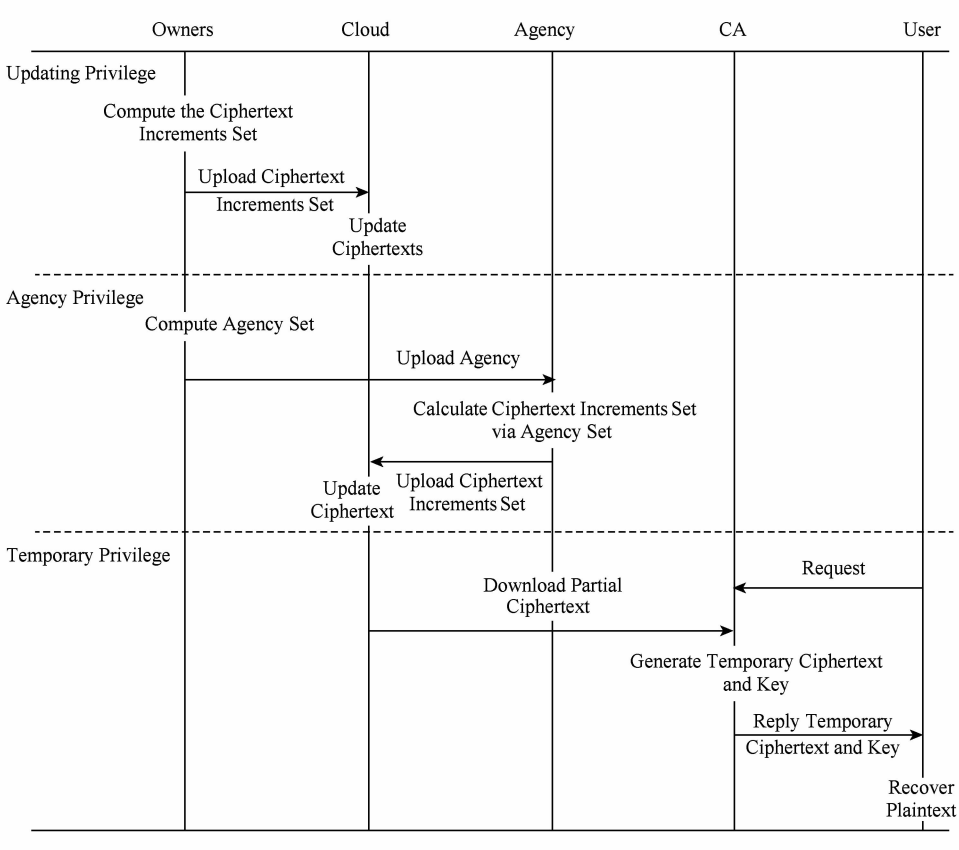


Fig. 2　Dynamic privilege.

图 2　动态授权

3.3　访问策略形态转换

访问结构的表示方法很多,其性能各不相同:单调布尔公式中只是简单地对 s 进行分割,计算简单,但表现力较差,且可扩展性不强;访问树表现力强,逻辑结构清晰,但需逐层进行计算,效率不高;LSSS 矩阵不仅具备丰富的表现力且计算效率相对较高.为使动态授权方法具备通用性,本文给出将单调布尔公式和访问树转换为 LSSS 矩阵的算法,再以 LSSS 矩阵为基础构建灵活的动态授权方法.该方法在不改变原有 shares 集合 S 的前提下,将原有的访问策略以其他形式表示出来,使其在逻辑上及数值上都是等效的.因此,通过该算法对访问策略进行转换时密文部分不需要进行更改.

3.3.1　访问树转换为 LSSS 矩阵

文献[1]对访问树给出了具体的定义及实现方法.在访问树中,若节点 $node$ 为非叶子节点(即 t-out-of-n 节点),则 $node$ 对应 1 个常数项为 $s(node)$ 的 $t-1$ 阶多项式 $f_{node}(x)$ 及 1 个插值 x_{node}:若 $node$ 为根节点,则 $s(node)$ 为选定的秘密 s,否则令 $s(node)=f_p(x_{node})$(p 为 $node$ 的父节点);若 $node$ 为叶子节点,则 $node$ 与 1 个属性相关,仅有 x_{node} 而没有 $f_{node}(x)$.

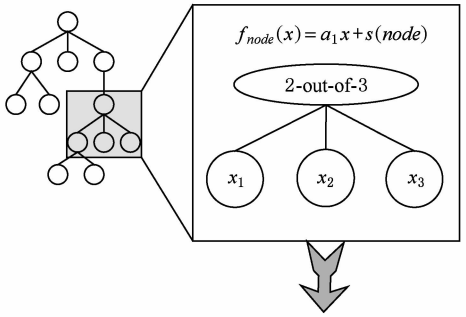
记非叶子节点 $node$ 对应的多项式为 $f_{node}(x)=a_{t-1}x^{t-1}+a_{t-2}x^{t-2}+\cdots+a_1x+s(node)$,$node$ 的每个子节点 $c_i(1\leqslant i\leqslant n)$ 对应 1 个插值 x_i($\forall x_i\neq x_j$,$i\neq j$),则该节点可表示为如下向量 $\boldsymbol{v}(node)$ 和矩阵 $\boldsymbol{M}(node)$:

$$\boldsymbol{v}(node)=(s(node),a_1,a_2,\cdots,a_{t-1}),\tag{4}$$

$$\boldsymbol{M}(node)=\begin{pmatrix}1 & x_1 & \cdots & x_1^{t-1}\\ 1 & x_2 & \cdots & x_2^{t-1}\\ \vdots & \vdots & & \vdots\\ 1 & x_n & \cdots & x_n^{t-1}\end{pmatrix}.\tag{5}$$

显然,$s(c_i)=f_{node}(x_i)=\boldsymbol{M}(node)_i\cdot\boldsymbol{v}(node)^{\mathrm{T}}$.由范德蒙矩阵的性质可知 $\boldsymbol{M}(node)$ 的任意 t 行可构成 1 个秩为 t 的子矩阵 $\boldsymbol{M}^*(node)$,故存在向量 $\boldsymbol{W}\in\mathbb{Z}^t$ 有 $\boldsymbol{W}\boldsymbol{M}^*(node)=(1,0,\cdots,0)$.因此 $\boldsymbol{M}(node)$ 可作为 $node$ 的节点矩阵.图 3 所示为 1 个节点矩阵转换示例.

本文根据文献[5]中提出的 LSSS 矩阵生成算法改进,得到递归算法 $TreeToLSSS(node,\varphi,\boldsymbol{M},\boldsymbol{V})$ 可将整个访问树转化为 LSSS 矩阵.其中,$node$ 表示访问树中的节点;φ 表示从整数集合到访问数中节点集合的映射函数;$\boldsymbol{M}$ 和 $\boldsymbol{V}$ 分别表示访问树对应的



$$\boldsymbol{M}(node)=\begin{pmatrix}1 & x_1\\ 1 & x_2\\ 1 & x_3\end{pmatrix},\quad \begin{aligned}&\boldsymbol{v}(node)=(s(node),a_1)\\ &s(c_i)=\boldsymbol{M}(node)_i\cdot\boldsymbol{v}(node)^{\mathrm{T}}\end{aligned}$$

Fig. 3 Tree node to node LSSS matrix.

图 3 访问树节点转换为 LSSS 矩阵及向量

LSSS 矩阵和随机向量,分别由多个节点矩阵 $\boldsymbol{M}(node)$ 和节点向量 $\boldsymbol{v}(node)$ 合并而成. 令 *root* 表示树的根节点,$\boldsymbol{M}_0=(1)$ 表示 LSSS 矩阵 $\boldsymbol{M}$ 的初始状态,$\boldsymbol{V}_0=(s)$ 则表示向量 $\boldsymbol{V}$ 的初始状态,φ_0 表示初始化的映射函数且 $\varphi_0(1)=root$. 通过执行算法 $TreeToLSSS(root,\varphi_0,\boldsymbol{M}_0,\boldsymbol{V}_0)$ 对整个访问树进行深度优先遍历,并生成对应的 LSSS 矩阵 $\boldsymbol{M}$ 及相应的向量 $\boldsymbol{V}$. 由算法 1 所生成的随机向量 $\boldsymbol{V}$ 及 LSSS 矩阵 $\boldsymbol{M}$ 的行向量 $\boldsymbol{M}_k$,$1\leqslant k\leqslant l$ 均可以看成分块向量. 记行向量 $\boldsymbol{M}_k=(1,\boldsymbol{M}_{k,n_1},\boldsymbol{M}_{k,n_2},\cdots,\boldsymbol{M}_{k,n_t})$,随机向量 $\boldsymbol{V}=(s,\boldsymbol{V}_{n_1},\boldsymbol{V}_{n_2},\cdots,\boldsymbol{V}_{n_t})$,其中 $n_i,i\in\mathbb{Z}_l^*$ 为访问树中的非叶子节点. 向量中每个分块 $\boldsymbol{V}_{node}$,$\boldsymbol{M}_{k,node}$ 对应树形结构中的 1 个非叶子节点 *node*:

$$\boldsymbol{V}_{node}=(a_1,a_2,\cdots,a_{t-1}),\tag{6}$$

$$\boldsymbol{M}_{k,node}=\begin{cases}(0,0,\cdots,0), & node\notin Anc(\varphi(k)),\\ (x_{c_i},x_{c_i}^2,\cdots,x_{c_i}^{t-1}), & node\in Anc(\varphi(k)),\end{cases}\tag{7}$$

其中,$Anc(\varphi(k))$ 表示 $\varphi(k)$ 祖先节点的集合,c_i 表示 *node* 的第 i 个子节点且 $c_i\in Anc(\varphi(k))\cup\varphi(k)$,节点多项式 $f_{node}(x)=a_{t-1}x^{t-1}+a_{t-2}x^{t-2}+\cdots+a_1+s(node)$,$a_m$,$1\leqslant m\leqslant t-1$ 为 $f_{node}(x)$ 中的非常数项系数. 本文中所定义的 LSSS 矩阵均具备此分块特征,该特征使得 LSSS 矩阵具备较好的可扩展性且易于管理. 同时,通过该算法转化所得 LSSS 矩阵与原有访问树在逻辑上及数值上都是等效的,其逻辑结构与 shares 集合都不需改变.

算法 1. $TreeToLSSS(node,\varphi,\boldsymbol{M},\boldsymbol{V})$.

输入:访问树 T 中的节点 *node*、矩阵 $\boldsymbol{M}\in\mathbb{Z}^{l\times m}$、向量 $\boldsymbol{V}\in\mathbb{Z}^m$、从 $\boldsymbol{M}$ 中行坐标到数树中节点的映射函数 φ;

输出:矩阵 $\boldsymbol{M}'\in\mathbb{Z}^{l\times m}$、向量 $\boldsymbol{V}'\in\mathbb{Z}^m$、映射函数 φ'.

If *node* 为叶子节点 do

  $\boldsymbol{V}'\leftarrow\boldsymbol{V}$;

  $\boldsymbol{M}'\leftarrow\boldsymbol{M}$;

  $\varphi'\leftarrow\varphi$;

Else

  $\boldsymbol{V}'\leftarrow(\boldsymbol{V},a_1,a_2,\cdots,a_{t-1})$; /* *node* 为 t-out-of-n 节点,$f_{node}(x)=\sum_{t'=1}^{t-1}a_{t'}x^{t'}+s(node)$ */

  $k\leftarrow1$;

  For all $j\in\{1,2,\cdots,l\}$ do

    If $\rho(j)=node$ do

      For all $i\in\{1,2,\cdots,n\}$ do

        $\boldsymbol{M}'_k\leftarrow(\boldsymbol{M}_j,x_{c_i},x_{c_i}^2,\cdots,x_{c_i}^{t-1})$; /* c_i 为 *node* 的子节点 */

        $\varphi'(k)\leftarrow c_i$;

        $k\leftarrow k+1$;

      End For

    Else

      $\boldsymbol{M}'_k\leftarrow(\boldsymbol{M}_j,0,\cdots,0)$; /* 填充 t 个 0 分量 */

      $\varphi'(k)\leftarrow\varphi(j)$;

      $k\leftarrow k+1$;

    End If

  End For

  For all $i\in\{1,2,\cdots,n\}$ do

    $(\boldsymbol{M}',\boldsymbol{V}',\varphi')\leftarrow TreeToLSSS(c_i,\varphi',\boldsymbol{M}',\boldsymbol{V}')$;

  End For

End If

Return $(\boldsymbol{M}',\boldsymbol{V}',\varphi')$.

3.3.2 单调布尔公式转换为访问树

在单调布尔公式中,门限结构($Th_{t,n}$)是由 AND($\wedge$)和 OR($\vee$)组合表示的,如:

$$Th_{2,3}(A,B,C)\leftrightarrow(A\wedge B)\vee(A\wedge C)\vee(B\wedge C).$$

因此,为简单起见,可假定单调的布尔公式中只存在 AND 和 OR 两种谓词逻辑且 AND 和 OR 均为二元操作,即任意单调的布尔公式均可看成 1 棵完全二叉树,如图 4 所示.

在单调布尔公式中,秘密 s 的划分如下:AND 谓词将对应的 s 分解为 2 个分量 s_1,s_2 分别赋予左右 2 子节点,且 $s_1+s_2=s$;OR 谓词则将对应的 s 直接赋值于其子节点,即 $s_1=s_2=s$. 将布尔公式对应的完全二叉树转换为访问树需要计算各节点的节点多项式和节点插值. 若 *node* 为或(OR)节点,节点对应的多项式函数为常函数 $f_{node}(x)=s(node)$;若

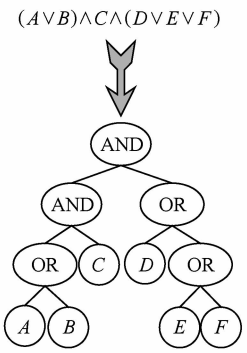


Fig. 4　Monotonous Boolean expression to complete binary tree.

图 4　单调布尔公式的完全二叉树表示

$node$ 为与(AND)节点,节点多项式记 $f_{node}(x)=ax+s(node)$,$node$ 的左子节点记为 l,右子节点记为 r,于是有方程组:

$$\begin{cases} ax_l+s(node)=s_1, \\ ax_r+s(node)=s_2. \end{cases} \tag{8}$$

该方程组必定存在有理数解,因此,在 1 个素数阶的整群 $\mathbb{Z}_p$ 中该方程能求得有效解.得到各节点对应的多项式及插值后则可根据算法 1 生成与该布尔公式对应的 LSSS 矩阵 $\boldsymbol{M}$ 及随机向量 $\boldsymbol{V}$.同样地,该转换不会改变访问策略的逻辑结构和 shares 集合.

3.4　基于 ABE 的访问控制机制

本文通过 ABE 为云存储体系构建访问控制机制,提供安全可靠的数据访问服务.基于 ABE 的访问控制系统中包括 4 个基本功能模块:$Setup$,$KeyGen$,$Encrypt$,$Decrypt$.$Setup$ 中,CA 及 AA 定义公钥 PK 及主钥 MK;$KeyGen$ 中,CA 及各 AA 为合法用户分发私钥 SK;$Encrypt$ 由数据所有者执行,以明文 m、公钥 PK 及访问策略 T 为输入,生成密文 CT 及 T 对应的 LSSS 矩阵;$Decrypt$ 由访问数据的用户执行,以用户私钥 SK、密文 CT 及其相关的 LSSS 矩阵为输入,当且仅当用户私钥对应属性集合为该 LSSS 矩阵确定的 1 个可满足集时,用户可还原明文.

1) $Setup(1^l)$.CA 根据安全参数 l 生成 p 阶的双线性群 G_0,G_1 和双线性映射 $e:G_0\times G_0\to G_1$.CA 选取 $g_1,g_2\in G_0$ 以及 $x,w\in\mathbb{Z}_p^*$ 并令 $MK_{\text{global}}=\{w,x\}$,生成全局公钥:

$$PK_{\text{global}}=\{g_2,P'=g_1,P''=g_1^x,Y=e(g_1,g_2)^w\}.$$

设系统中共有 N 个属性授权机构,AA_i 表示第 i 个属性授权机构,$S_i=\{A_{i,j}\}_{1\leqslant j\leqslant n_i}$ 为 AA_i 定义的属性集合,其中 $A_{i,j}$ 表示 AA_i 定义的第 j 个属性且 $\#S_i=n_i$.每个 AA_i 选取整数集合 $MK_i=\{a_{i,j}\in\mathbb{Z}_p\}_{1\leqslant j\leqslant n_i}$ 并计算:

$$PK_i=\{P_{i,j}=(P'')^{a_{i,j}}\}_{1\leqslant j\leqslant n_i},$$

其中,$a_{i,j}$ 与 $P_{i,j}$ 分别表示属性 $A_{i,j}$ 对应的主钥及公钥.于是可得到完整的系统主钥 MK 及公钥 PK:

$$MK=\{MK_{\text{global}},MK_i\}_{1\leqslant i\leqslant N},$$
$$PK=\{PK_{\text{global}},PK_i\}_{1\leqslant i\leqslant N}.$$

2) $KeyGen(u,U_s,MK)$.用户选取随机值 $s_u\in\mathbb{Z}_p$ 及 $u,p\in G_0$,计算 $U=up^{s_u}$ 并将元组 $\langle U,p\rangle$ 发送给 CA 及各 AA 申请全局私钥及属性私钥.CA 返回给用户如下元组:

$$TK_{\text{global}}=\langle UID'=g_2^wU^{-x},UID''=p^x\rangle.$$

AA_i 生成元组 TK_i 并发送给 CA:

$$TK_i=\langle\bar{u}_{i,j}=U^{a_{i,j}},\tilde{u}_{i,j}=p^{a_{i,j}}\rangle_{A_{i,j}\in U_s\cap S_i}.$$

其中,U_s 表示用户的属性集合.CA 根据 TK_i 计算 TK_{att} 并返回给用户:

$$TK_{\text{att}}=\{u'=\bar{u}^x,u''=\tilde{u}^x\}_{A_{i,j}\in U_s},$$

用户计算:

$$D_{\text{global}}=UID'(UID'')^{s_u}=g_2^wu^{-x}, \tag{9}$$
$$D_{i,j}=u'(u'')^{-s_u}=u^{a_{i,j}x}. \tag{10}$$

于是,得到用户完整的私钥

$$SK_u=\{D_{\text{global}},u,D_{i,j}\}_{A_{i,j}\in U_s}.$$

3) $Encrypt(\mu,\boldsymbol{M},\boldsymbol{V},\rho,PK)$.数据所有者对明文 μ 进行加密时,首先计算 $C_0=\mu Y^s$,$C'=g_1^s$ 并根据算法 1 生成 LSSS 矩阵 $\boldsymbol{M}\in\mathbb{Z}^{l\times m}$,向量 $\boldsymbol{V}\in\mathbb{Z}^m$,同时定义行坐标到系统属性集合的映射函数 $\rho:\mathbb{Z}_l^*\to S_1\cup S_2\cup\cdots\cup S_N$.下一步,数据所有者计算:

$$\lambda_k=\boldsymbol{M}_k\boldsymbol{V}^{\mathrm{T}}, \tag{11}$$
$$\bar{C}_{\rho(k)}=(P')^{r_k}=g_1^{r_k}, \tag{12}$$
$$C_{\rho(k)}=(P'')^{\lambda_k}(P_{\rho(k)})^{r_k}=g_1^{x\lambda_k+r_ka_{\rho(k)}x}, \tag{13}$$

其中,$\boldsymbol{M}_k$ 表示矩阵 $\boldsymbol{M}$ 的第 k 行,$\rho(k)$ 表示 $\boldsymbol{M}_k$ 关联的一个属性.完整的密文表示为

$$CT=\{C_0,C',C_k,\bar{C}_k\}_{1\leqslant k\leqslant l}.$$

4) $Decrypt(CT,SK_u)$.若用户属性集合满足密文的访问策略,则必存在集合 $\{\rho(k)\mid 1\leqslant k\leqslant l\}$ 的 1 个子集 I_u,有 $I_u\subset U_s$ 且:

$$\sum_{\rho(k)\in I_u}\omega_k\boldsymbol{M}_k=(1,0,\cdots,0).$$

于是有:

$$\frac{C_0}{e(D_{\text{global}},C')}\prod_{\rho(k)\in I_u}\left(\frac{e(u,C_k)}{e(D_{\rho(k)},\bar{C}_k)}\right)^{-\omega_k}=$$
$$\frac{\mu e(g_1,g_2)^{ws}}{e(g_2^wu^{-x},g_1^s)}\prod_{\rho(k)\in I_u}\left(\frac{e(u,g_1^{\lambda_kx+r_ka_{\rho(k)}x})}{e(u^{a_{\rho(k)}x},g_1^{r_k})}\right)^{-\omega_k}=$$
$$\frac{\mu}{e(u,g_1)^{-sx}}\prod_{\rho(k)\in I_u}e(u,g_1)^{-\omega_k\lambda_kx}=\mu.$$

3.5 动态授权

灵活高效的动态授权方法可增强云存储体系中数据的易管理性,并有效提高系统的整体运行效率.本文根据授权操作执行者的不同提出3类动态授权方法:数据所有者的更新授权、代理三方的代理授权以及CA授予的临时授权.3类动态授权方法适用于不同的应用场景.同时,由于树形的访问策略在表示上逻辑结构更为清晰、便于理解,本文的动态授权方法用树形结构描述访问策略的结构更改,再对相应的LSSS矩阵进行更新,从而实现访问策略的更新.

3.5.1 更新授权

从功能上可将动态策略更改分为2种类型:策略结构的更改(即对LSSS矩阵 $\boldsymbol{M}$ 进行更改)和策略中属性的更改(即对映射函数 ρ 进行更改).对策略结构进行更改时,首先要对矩阵 $\boldsymbol{M}$ 及随机向量 $\boldsymbol{V}$ 进行调整.记原LSSS矩阵为 $\boldsymbol{M}$,其行向量表示为 $\boldsymbol{M}_k=(1,\boldsymbol{M}_{k,n_1},\boldsymbol{M}_{k,n_2},\cdots,\boldsymbol{M}_{k,n_l})$,随机向量表示为 $\boldsymbol{V}=(s,\boldsymbol{V}_{n_1},\boldsymbol{V}_{n_2},\cdots,\boldsymbol{V}_{n_l})$,修改后的矩阵为 $\boldsymbol{M}'$,其行向量为表示为 $\boldsymbol{M}'_k=(1,\boldsymbol{M}'_{k,n_1},\boldsymbol{M}'_{k,n_2},\cdots,\boldsymbol{M}'_{k,n_l})$,随机向量表示为 $\boldsymbol{V}'=(s,\boldsymbol{V}'_{n_1},\boldsymbol{V}'_{n_2},\cdots,\boldsymbol{V}'_{n_l})$(如3.3.1节中所定义).更改LSSS矩阵时计算出矩阵中每个行向量所对应的增量 $\Delta\lambda_k=\sum_{i\in\boldsymbol{Z}_t}(\boldsymbol{M}'_{k,n_i}(\boldsymbol{V}'_{n_i})^{\mathrm{T}}-\boldsymbol{M}_{k,n_i}(\boldsymbol{V}_{n_i})^{\mathrm{T}})$,再根据增量集合 $\{\Delta\lambda_k\mid 1\leqslant k\leqslant l,\Delta\lambda\neq 0\}$ 对密文的局部进行更新即可实现对其访问策略的更改.由于矩阵 $\boldsymbol{M}$ 和随机向量 $\boldsymbol{V}$ 具备明显的分块特性,因此访问策略的结构的更改可以"块"为单位进行.本文定义4个以块为单位执行策略更改的函数:$Update_{\mathrm{Th}}$,$Update_{\mathrm{path}}$,$Remove_{\mathrm{subtree}}$ 和 Add_{subtree},对LSSS矩阵的任何更改都可以通过调用这4个基本函数实现.

1) $Update_{\mathrm{Th}}(n_v,t_v)$. 该函数将访问策略中的1个 t-out-of-n 节点 n_v 变为 t_v-out-of-n,$\forall\boldsymbol{M}_k,k\in\mathbb{Z}$ 及 $\boldsymbol{V}$ 中仅需要修改 $\boldsymbol{M}_{k,n_v}$ 及 $\boldsymbol{V}_{n_v}$ 部分.记为

$$\boldsymbol{V}_{n_v}=(a_1,a_2,\cdots,a_{t-1}),\tag{14}$$

$$\boldsymbol{M}_{k,n_v}=\begin{cases}(0,0,\cdots,0),\ n_v\notin Anc(\varphi(k));\\(x_{c_i},x_{c_i}^2,\cdots,x_{c_i}^{t-1}),\ n_v\in Anc(\varphi(k)).\end{cases}\tag{15}$$

若 $t_v>t$,则有:

$$\boldsymbol{V}'_{n_v}=(a_1,a_2,\cdots a_{t-1},a_t,\cdots,a_{t_v-1}),\tag{16}$$

$$\boldsymbol{M}'_{k,n_v}=\begin{cases}(0,0,\cdots,0),\ n_v\notin Anc(\varphi(k));\\(x_{c_i},x_{c_i}^2,\cdots,x_{c_i}^{t-1},\cdots,x_{c_i}^{t_v-1}),\\\quad n_v\in Anc(\varphi(k)).\end{cases}\tag{17}$$

于是有:

$$\Delta\lambda_k=\begin{cases}0,\ n_v\notin Anc(\varphi(k)).\\\sum_{j=t_v}^{t}a_jx_{c_i}^j,\ n_v\in Anc(\varphi(k)).\end{cases}\tag{18}$$

若 $t_v<t$,则有:

$$\boldsymbol{V}'_{n_v}=(a_1,a_2,\cdots,a_{t_v-1}),\tag{19}$$

$$\boldsymbol{M}'_{k,n_v}=\begin{cases}(0,0,\cdots,0),\ n\notin Anc(\varphi(k)),\\(x_{c_i},x_{c_i}^2,\cdots,x_{c_i}^{t_v-1}),\ n\in Anc(\varphi(k)),\end{cases}\tag{20}$$

且:

$$\Delta\lambda_k=\begin{cases}0,\ n_v\notin Anc(\varphi(k));\\\sum_{j=t}^{t_v}-a_jx_{c_i}^j,\ n_v\in Anc(\varphi(k)).\end{cases}\tag{21}$$

2) $Updata_{\mathrm{path}}(r_v,p_v)$. 该函数将访问策略中1棵以 r_v 为根节点的子树更改为节点 p_v 的子树.记分段向量 $\boldsymbol{M}_{r_v}=(1,\boldsymbol{M}_{r_v,n_1},\boldsymbol{M}_{r_v,n_2},\cdots,\boldsymbol{M}_{r_v,n_l})$ 且:

$$\boldsymbol{M}_{r_v,n_k}=\begin{cases}(0,0,\cdots,0),\ n_k\notin Anc(r_v)\bigcup\{r_v\},\\(x_{c_i},x_{c_i}^2,\cdots,x_{c_i}^{t_{n_k}-1}),\\\quad n_k\in Anc(r_v)\bigcup\{r_v\},\end{cases}\tag{22}$$

其中,c_i 表示 n_k 下1个子节点且 $c_i\in Anc(p_v)\bigcup\{r_v\}$,$1\leqslant k\leqslant l$.同理,记 $\boldsymbol{M}'_{r_v}=(1,\boldsymbol{M}'_{r_v,n_1},\boldsymbol{M}'_{r_v,n_2},\cdots,\boldsymbol{M}'_{r_v,n_l})$,且有:

$$\boldsymbol{M}'_{r_v,n_k}=\begin{cases}(0,0,\cdots,0),\ n_k\notin Anc(p_v),\\(x_{c_i},x_{c_i}^2,\cdots,x_{c_i}^{t_{n_k}-1}),\ n_k\in Anc(p_v),\\(x_{r_v},x_{r_v}^2,\cdots,x_{r_v}^{t_{p_v}-1}),\ n_k=p_v,\end{cases}\tag{23}$$

其中,x_{r_v} 表示为节点 r_v 重新分配的节点插值.再记 $\Delta\boldsymbol{M}_{r_v}=\boldsymbol{M}'_{r_v}-\boldsymbol{M}_{r_v}$,$\Delta\lambda_{r_v}=\Delta\boldsymbol{M}_{r_v}\boldsymbol{V}^{\mathrm{T}}$.若 $\varphi(k)$ 为该子树中的叶子节点,则令:

$$\boldsymbol{M}'_k=\boldsymbol{M}'_k+\Delta\boldsymbol{M}_{r_v},$$
$$\Delta\lambda_k=\Delta\lambda_{r_v}.$$

3) $Remove_{\mathrm{subtree}}(r_v)$. 该函数将删除访问策略树形结构中以节点 r_v 为根节点的1棵子树.由于矩阵中各分块相对独立,删除矩阵中某些分块并不会对其他分块产生影响.因此,删除子树时可直接对矩阵的某些行列进行删除:删除行向量 $\{\boldsymbol{M}_{k'}\mid\varphi(k')\in sub(r_v)\}$ 和列分块 $\{\boldsymbol{M}_{k,node}\mid node\in sub(r_v),\varphi(k)\notin sub(r_v)\}$ 及 $\{\boldsymbol{V}_{node}\mid node\in sub(r_v)\}$,其中 $sub(r_v)$ 表示该子树中所有节点的集合.显然,所有分块 $\{\boldsymbol{M}_{k,node}\mid node\in sub(r_v),\varphi(k)\notin sub(r_v)\}$ 均为零向量,于是有 $\Delta\lambda_k=0,\varphi(k)\notin sub(r_v)$,因此 $\{\Delta\lambda_k\mid\Delta\lambda\neq 0\}=\varnothing$.

4) $Add_{\mathrm{subtree}}(r_v,p_v)$. 该函数将在树形结构中的指定节点 p_v 下插入1棵以 r_v 为根节点的子树.记该子树的LSSS矩阵为 $\boldsymbol{M}(r_v)$,随机向量为 $\boldsymbol{V}_{r_v}$,映射函数为 φ_{r_v},S 为原树中非叶子节点的集合,S_{r_v} 为所增加子树中的非叶子节点集合.为 r_v 选取插值 x_{r_v},记向量 $\boldsymbol{V}_{p_v}=(1,\boldsymbol{M}_{p_v,n_1},\boldsymbol{M}_{p_v,n_2},\cdots,\boldsymbol{M}_{p_v,n_l})$ 且:

$$\boldsymbol{M}'_{p_v,n_k}=\begin{cases}(0,0,\cdots,0),\ n_k\notin Anc(p_v);\\(x_{c_i},x_{c_i}^2,\cdots,x_{c_i}^{t_{n_k}-1}),\ n_k\in Anc(p_v);\\(x_{r_v},x_{r_v}^2,\cdots,x_{r_v}^{t_{r_v}-1}),\ n_k=p_v.\end{cases}\tag{24}$$

再令 $\Delta\lambda_{r_\upsilon}=\boldsymbol{M}_{p_\upsilon}\boldsymbol{V}^{\mathrm{T}}$,$\Delta\lambda_{k'}=\widetilde{\boldsymbol{M}}(r_\upsilon)_{k'}\widetilde{\boldsymbol{V}}_{r_\upsilon}^{\mathrm{T}}$,$\boldsymbol{M}(p_\upsilon)=(\boldsymbol{V}_{p_\upsilon},\cdots,\boldsymbol{V}_{p_\upsilon})^{\mathrm{T}}$,并生成 LSSS 矩阵 $\boldsymbol{M}'$及随机向量 $\boldsymbol{V}'$:

$$\boldsymbol{M}'=\begin{pmatrix}\boldsymbol{M} & \boldsymbol{O}\\ \boldsymbol{M}(p_\upsilon) & \widetilde{\boldsymbol{M}}(r_\upsilon)\end{pmatrix},\tag{25}$$

$$\boldsymbol{V}'=(\boldsymbol{V},\widetilde{\boldsymbol{V}}_{r_\upsilon}),\tag{26}$$

其中,$\boldsymbol{O}$ 表示全零矩阵;$\widetilde{\boldsymbol{M}}_{r_\upsilon}$,$\widetilde{\boldsymbol{V}}_{r_\upsilon}$ 分别表示 $\boldsymbol{M}_{r_\upsilon}$,$\boldsymbol{V}_{r_\upsilon}$ 删除第 1 列分量后的子矩阵和子向量. 于是,有:

$$\Delta\lambda_{\rho(k)}=\begin{cases}\Delta\lambda_{r_\upsilon}+\Delta\lambda_{k'},\ \varphi(k)\in sub(r_\upsilon),\\ \varphi_{r_\upsilon}(k')=\varphi(k);\\ 0,\ \varphi(k)\notin sub(r_\upsilon).\end{cases}\tag{27}$$

通过 4 个基本函数完成对 LSSS 矩阵的动态更改后,获取增量集合 $\{\Delta\lambda_{\rho(k)}\mid\varphi(k)\in S_{\mathrm{update}}\}$. 其中,$S_{\mathrm{update}}$ 表示改动的叶子节点集合,计算密文增量集合 $\Delta'=\{\Delta C'_{\rho(k)}=(P'')^{\Delta\lambda_{\rho(k)}}\}_{\rho(k)\in S_{\mathrm{update}}}$. 由于可能存在多个 $\Delta\lambda_{\rho(k)}$ 相等的情况,此时仅需记为 1 个 $\Delta C'_{\rho(k)}$ 即可. 云存储服务器接收到 Δ'后计算:

$$\Delta\overline{C}_{\rho(k)}=(P')^{\Delta r_{\rho(k)}},\tag{28}$$

$$\Delta C_{\rho(k)}=\Delta C'_{\rho(k)}(P_{\rho(k)})^{\Delta r_{\rho(k)}},\tag{29}$$

其中,$\varphi(k)\in S_{\mathrm{update}}$. 再对密文进行更新:

$$\overline{C}_{\rho(k)}=\overline{C}_{\rho(k)}\Delta\overline{C}_{\rho(k)},\tag{30}$$

$$C_{\rho(k)}=C_{\rho(k)}\Delta C_{\rho(k)}.\tag{31}$$

属性更改较为简单:将 $\rho(k)$更改为 $\rho'(k)$,数据所有者计算 $\Delta\overline{C}_{\rho'(k)}=(P')^{\Delta r_{\rho'(k)}}$,$\Delta C_{\rho'(k)}=(P_{\rho(k)})^{-r_{\rho(k)}}(P_{\rho'(k)})^{r_{\rho(k)}+\Delta r_{\rho'(k)}}$ 并上传到云存储服务器上. 服务器计算 $\overline{C}_{\rho'(k)}=\overline{C}_{\rho(k)}\Delta\overline{C}_{\rho'(k)}$,$C_{\rho'(k)}=C_{\rho(k)}\Delta C_{\rho'(k)}$.

3.5.2　代理授权

若局部的访问策略需要频繁变动而数据所有者无法及时对策略进行更新,则可将该访问策略委托给第三方(代理方)代为管理. 数据所有者在树形结构中指定一个叶子节点 n_υ 作为代理节点,代理服务器可在代理节点下插入代理子树,并负责对代理子树进行管理,如图 5 所示:

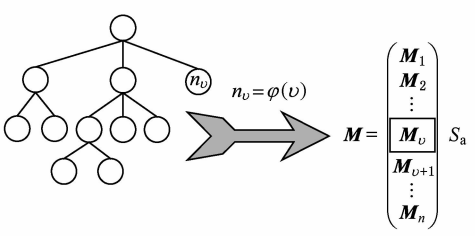


Fig. 5　Agency node/row vector in access policy.

图 5　访问策略中的代理节点/行向量

数据所有者制定访问策略后,根据策略生成 LSSS 矩阵 $\boldsymbol{M}$,随机向量 $\boldsymbol{V}$ 和映射函数 φ,并在访问策略中指定 1 个代理叶子节点 $n_\upsilon=\varphi(\upsilon)$. 下一步,数据所有者生成局部密文 $CT=\{C_0,C',C_k,\overline{C}_k\}_{k\neq\upsilon}$. 记 n_υ 对应的代理行向量为 $\boldsymbol{M}_\upsilon$,且有 $\lambda_{n_\upsilon}=\boldsymbol{M}_{n_\upsilon}\boldsymbol{V}^{\mathrm{T}}$. 数据所有者选取 1 个代理属性集合 S_{n_υ},计算代理授权集合 $S_{\mathrm{a}}=\{\overline{C}_{i,j}=g_1^{r_{i,j}},C_{i,j}=g_1^{x\lambda_{n_\upsilon}+r_{i,j}a_{i,j}x}\}_{A_{i,j}\in S_{n_\upsilon}}$ 并将 S_{a} 发送给代理方. 代理方以 S_{n_υ} 为基础制定代理子树 T_{sub},并生成相应的代理子矩阵 $\boldsymbol{M}(T_{\mathrm{sub}})\in\mathbb{Z}^{l_{T_{\mathrm{sub}}}\times m_{T_{\mathrm{sub}}}}$、随机向量 $\boldsymbol{V}(T_{\mathrm{sub}})\in\mathbb{Z}^{m_{T_{\mathrm{sub}}}}$ 和映射函数 $\varphi_{T_{\mathrm{sub}}}$. 代理子树中各叶子节点增量记为 $\Delta\lambda_{k'}=\widetilde{\boldsymbol{M}}(T_{\mathrm{sub}})_{k'}\widetilde{\boldsymbol{V}}(T_{\mathrm{sub}})^{\mathrm{T}}$,$1\leqslant k'\leqslant l_{T_{\mathrm{sub}}}$,其中,$\boldsymbol{M}(T_{\mathrm{sub}})$表示 $\widetilde{\boldsymbol{M}}(T_{\mathrm{sub}})$删除第 1 列分量后的子矩阵,$\widetilde{\boldsymbol{V}}(T_{\mathrm{sub}})$表示 $\boldsymbol{V}(T_{\mathrm{sub}})$删除第 1 列分量后的子向量. 更新访问策略,得到 LSSS 矩阵 $\boldsymbol{M}'$和随机向量 $\boldsymbol{V}'$:

$$\boldsymbol{M}'=\begin{pmatrix}\boldsymbol{M}^* & \boldsymbol{O}\\ \boldsymbol{E}\boldsymbol{M}_{n_\upsilon} & \widetilde{\boldsymbol{M}}_{T_{\mathrm{sub}}}\end{pmatrix},\tag{32}$$

$$\boldsymbol{V}'=(\boldsymbol{V},\widetilde{\boldsymbol{V}}_{T_{\mathrm{sub}}}),\tag{33}$$

其中,$\boldsymbol{M}^*$ 为矩阵 $\boldsymbol{M}$ 删除代理行向量 $\boldsymbol{M}_\upsilon$ 后得到的子矩阵,$\boldsymbol{E}$ 为全 1 列向量. 记 $L(T_{\mathrm{sub}})$为 T_{sub}中叶子节点的集合,代理方根据 $L(T_{\mathrm{sub}})$计算增量集合 $\Delta=\{\Delta\overline{C}_{\varphi_{T_{\mathrm{sub}}}(k)},\Delta C_{\varphi_{T_{\mathrm{sub}}}(k)}\}_{\varphi_{T_{\mathrm{sub}}}(k)\in L(T_{\mathrm{sub}})}$,其中:

$$\Delta\overline{C}_{\varphi_{T_{\mathrm{sub}}}(k)}=(P')^{\Delta r_{\varphi_{T_{\mathrm{sub}}}(k)}},\tag{34}$$

$$\Delta C_{\varphi_{T_{\mathrm{sub}}}(k)}=(P'')^{\Delta\lambda_{\varphi_{T_{\mathrm{sub}}}(k)}}(P_{\varphi_{T_{\mathrm{sub}}}(k)})^{\Delta r_{\varphi_{T_{\mathrm{sub}}}(k)}}.\tag{35}$$

进一步,根据 Δ 计算元组 $Agent=\{\overline{C}_{\varphi_{T_{\mathrm{sub}}}(k)},C_{\varphi_{T_{\mathrm{sub}}}(k)}\}_{\varphi_{T_{\mathrm{sub}}}(k)\in L(T_{\mathrm{sub}})}$,其中,

$$\overline{C}_{\varphi_{T_{\mathrm{sub}}}(k)}=\overline{C}_{\rho_{T_{\mathrm{sub}}}(k)}\Delta\overline{C}_{\varphi_{T_{\mathrm{sub}}}(k)},\tag{36}$$

$$C_{\varphi_{T_{\mathrm{sub}}}(k)}=C_{\rho_{T_{\mathrm{sub}}}(k)}\Delta C_{\varphi_{T_{\mathrm{sub}}}(k)}.\tag{37}$$

其中,$\rho_{T_{\mathrm{sub}}}$表示由 $\boldsymbol{M}(T_{\mathrm{sub}})$的行坐标到集合 S_{n_υ}的映射函数. 代理方将元组 $Agent$ 传送给云端,云端服务器将 $Agent$ 加入到原密文中,即完成了代理方对访问策略的更新. 若要对代理子树 T_{sub}的结构作调整,同样可通过调用 $Update_{\mathrm{Th}}$,$Update_{\mathrm{path}}$,$Remove_{\mathrm{subtree}}$,$Add_{\mathrm{subtree}}$等函数实现.

3.5.3　临时授权

临时授权是 CA 在紧急情况下授予用户的一次性访问权限. 记有密文 $CT=\{C_0,C',C_k,\overline{C}_k\}_{1\leqslant k\leqslant l}$. 用户在向 CA 发出对 CT 的临时访问请求时,首先选取随机参数 $g_{\mathrm{tmp}}\in G_0$,$s_{\mathrm{tmp}}\in\mathbb{Z}_p$,并将元组$\{g_{\mathrm{tmp}},D_{\mathrm{tmp}}=ug_{\mathrm{tmp}}^{s_{\mathrm{tmp}}}\}$发送给 CA. CA 收到用户请求后,从云存储服务器中获取密文参数 C_0,选取随机参数 x_{tmp},$\Delta w\in\mathbb{Z}_p$,并计算:

$$\widetilde{C}_0=C_0e(g_2^{\Delta w},C')=\mu e(g_1,g_2)^{(w+\Delta w)s},\tag{38}$$

$$\widetilde{C}'=(C')^{1/x_{\mathrm{tmp}}}=g_1^{s/x_{\mathrm{tmp}}},\tag{39}$$

$$\widetilde{D}=(D_{\mathrm{tmp}}g_2^{w+\Delta w})^{x_{\mathrm{tmp}}}=g_2^{(w+\Delta w)x_{\mathrm{tmp}}}u^{-x_{\mathrm{tmp}}}g_{\mathrm{tmp}}^{s_{\mathrm{tmp}}x_{\mathrm{tmp}}},\tag{40}$$

$$\widetilde{g}_{\mathrm{tmp}}=g_{\mathrm{tmp}}^{x_{\mathrm{tmp}}},\tag{41}$$

其中,$\widetilde{C}_0$ 为临时密文,$\{\widetilde{C}',\widetilde{D},\widetilde{g}_{\text{tmp}}\}$为该用户的会话密钥. CA 将元组$\langle\widetilde{C}_0,\widetilde{C}',\widetilde{D},\widetilde{g}_{\text{tmp}}\rangle$返回给用户,用户通过会话密钥还原零时密文:

$$\frac{\widetilde{C}_0}{e(\widetilde{C}',\widetilde{D}\,\widetilde{g}_{\text{tmp}}^{-x_{\text{tmp}}})e(C',u)}=\frac{\mu e(g_1,g_2)^{(w+\Delta w)s}}{e(g_1^{s/x_{\text{tmp}}},g_2^{(w+\Delta w)x_{\text{tmp}}}u^{-x_{\text{tmp}}})e(g_1,u)^s}=\mu.$$

在同时对多个用户授予临时访问权限时,CA 只需计算一次临时密文 $\widetilde{C}_0$,从而降低其计算开销.

3.6 安全性证明

假设有攻击者 A 在本文定义的选择性游戏(安全模型)中具备不可忽略的优势 ε,则有模拟器 B 可在解决 q-parallel BDHE 假设时具备不可忽略的优势 $\varepsilon/2$.

1) 初始化. 模拟器 B 获得 1 个 q-parallel BDHE 元组,元组中参数 $\eta=\eta_v$ 且 v 服从伯努利分布 Bern(0.5). 攻击者给出 1 个访问结构对应的 LSSS 矩阵 $\boldsymbol{M}^*\in\mathbb{Z}_p^{l^*\times n^*}$ 及相关映射函数 ρ^*,其中 $n^*\leqslant q$.

2) 建立. 模拟器运行算法 $Setup(1^t)$ 生成系统参数. 令 $g_1=g_2=g$,$w=a^{q+1}$,$x=ax'$,$x'\in\mathbb{Z}$,于是有 $Y=e(g,g)^{a^{q+1}}=e(g^a,g^{a^q})$且 $P'=g$,$P''=(g^a)^{x'}$. 对每个属性 $A_{i,j}$,令:

$$a_{i,j}x=a'_{i,j}+\sum_{l\in S_{i,j}}\sum_{n=1}^{n^*}a^nM_{l,n}/b_l,$$

于是有:

$$P_{i,j}=g^{a'_{i,j}}\prod_{l\in S_{i,j}}\prod_{n=1}^{n^*}g^{a^nM_{l,n}/b_l}. \tag{42}$$

其中,$S_{i,j}$表示所有满足$\rho^*(k)=A_{i,j}$的行索引 k 的集合,$M_{l,n}$为 $\boldsymbol{M}^*$ 中第 l 行第 n 列的分量. 若 $S_{i,j}=\varnothing$,则有 $P_{i,j}=g^{a_{i,j}}$.

3) 阶段 1. 在此阶段中,攻击者 A 选取 1 个属性集合 U_s 并发送给模拟器 B 进行私钥查询,若 U_s 不能满足所提交的访问策略,模拟器为攻击者生成相应私钥. 选取 1 个向量 $\boldsymbol{w}=(w_1,w_2,\cdots,w_{n^*})$,其中 $w_1=1$ 且对任意 $\rho(l)\in U_s$ 有 $\boldsymbol{w}\boldsymbol{M}_l^*=0$. 根据 LSSS 矩阵的定义,若 U_s 不满足矩阵 $\boldsymbol{M}^*$ 则必定存在这样的向量. 用户随机选取 $u'\in\mathbb{Z}_p$,并获取参数 u:

$$u=g^{u'+\sum_{n=1}^{n^*}a^{q+1-n}w_n/x'}. \tag{43}$$

由于 U_s 不能满足访问策略,因此必定存在 1 组 CA 及各 AA 根据所选参数为用户生成全局私钥 D' 及属性私钥 $D_{i,j}$:

$$D'=g^{a^{q+1}}(g^{u'+\sum_{n=1}^{n^*}a^{q+1-n}w_n/x'})^{-ax'}=g^{-au'x'}\prod_{n=2}^{n^*}g^{-a^{q+2-n}w_n}, \tag{44}$$

$$D_{i,j}=(g^{u'+\sum_{n=1}^{n^*}a^{q+1-n}w_n/x'})^{(a'_{i,j}+\sum_{l\in S_i}\sum_{n=1}^{n^*}a^nM_{l,n}/b_l)}=g^{u'a'_{i,j}}g^{a'_{i,j}/x'\sum_{n=1}^{n^*}a^{q+1-n}w_n}g^{u'\sum_{l\in S_i}\sum_{n=1}^{n^*}a^nM_{l,n}/b_l}\times(g^{\sum_{n=1}^{n^*}a^{q+1-n}w_n/x'\sum_{l\in S_i}\sum_{n=1}^{n^*}a^nM_{l,n}/b_l})=g^{u'a'_{i,j}}g^{a'_{i,j}/x'\sum_{n=1}^{n^*}a^{q+1-n}w_n}g^{u'\sum_{l\in S_i}\sum_{n=1}^{n^*}a^nM_{l,n}/b_l}\times g^{\sum_{l\in S_i}\sum_{n=1}^{n^*}\sum_{n'=1}^{n^*}a^{q+1+n'-n}w_nM_{l,n'}/x'b_l}. \tag{45}$$

由于:

$$\sum_{l\in S_i}\sum_{n=1,n'=n}^{n^*}a^{q+1+n'-n}w_nM_{l,n'}/x'b_l=a^{q+1}\sum_{l\in S_i}\sum_{n=1}^{n^*}w_nM_{l,n}=0, \tag{46}$$

因此有:

$$D_{i,j}=g^{u'a'_{i,j}}g^{a'_{i,j}/x'\sum_{n=1}^{n^*}a^{q+1-n}w_n}g^{u'\sum_{l\in S_i}\sum_{n=1}^{n^*}a^nM_{l,n}/b_l}\times g^{\sum_{l\in S_i}\sum_{n=1}^{n^*}\sum_{n'\neq n}a^{q+1+n'-n}w_nM_{l,n'}/x'b_l}. \tag{47}$$

由于 D'及 $D_{i,j}$中不包含未知项 $g^{a^{q+1}}$,因此可简单计算出.

4) 挑战. 在这个阶段,模拟器 B 生成挑战密文. 攻击者 A 给出 2 个等长明文 μ_0,μ_1 并发送个模拟器. 模拟器抛掷硬币 b,并计算 $C_0=\mu_bT$. 选取随机数 $r'_k\in\mathbb{Z}$,$1\leqslant k\leqslant l^*$ 且令 $r_k=r'_k-sb_kx'$. 令$C'=g^s$ 且向量 $\boldsymbol{V}=(s,sa+v'_2,sa^2+v'_3,\cdots,sa^{n^*-1}+v'_{n^*})$,其中 $v'_n\in\mathbb{Z}$,$2\leqslant n\leqslant n^*$. 于是有

$$\lambda_k=s\sum_{n=1}^{n^*}a^{n-1}M_{k,n}+\sum_{n=2}^{n^*}v_nM_{k,n}. \tag{48}$$

由此,可得:

$$C_{\rho^*(k)}=g^{x\lambda_k+r_ka_{\rho^*(k)}x}=g^{x'(s\sum_{n=1}^{n^*}a^nM_{k,n}+\sum_{n=2}^{n^*}av_nM_{k,n})}\times g^{(r'_k-sb_kx')(a'_{\rho^*(k)}+\sum_{l\in S_k}\sum_{n=1}^{n^*}a^nM_{l,n}/b_l)}=g^{x's\sum_{n=1}^{n^*}a^nM_{k,n}}g^{x'\sum_{n=2}^{n^*}av_nM_{k,n}}g^{r'_ka'_{\rho^*(k)}+r'_k\sum_{l\in S_k}\sum_{n=1}^{n^*}a^nM_{l,n}/b_l}\times g^{-sb_kx'a'_{\rho^*(k)}}g^{-x's\sum_{n=1}^{n^*}a^nM_{k,n}}g^{-x'\sum_{l\in S_k-(k)}\sum_{n=1}^{n^*}M_{l,n}a^nsb_k/b_l}=g^{x'\sum_{n=2}^{n^*}av_nM_{k,n}}g^{r_{k'}a'_{i,j}+r_{k'}\sum_{l\in S_k}\sum_{n=1}^{n^*}a^nM_{l,n}/b_l}\times g^{-sb_kx'a'_{\rho^*(k)}}g^{-x'\sum_{l\in S_k-(k)}\sum_{n=1}^{n^*}M_{l,n}a^nsb_k/b_l}, \tag{49}$$

$$\overline{C}_{\rho^*(k)}=g^{r_k}. \tag{50}$$

于是,得到完整的密文$\{C_0, C', \overline{C}_{\rho^*(k)}, C_{\rho^*(k)}\}_{1\leqslant k\leqslant l^*}$.

5) 阶段 2.重复阶段 1.

猜测.攻击者给出对 b 的猜测 b'.若 $b'=b$,模拟器给出 $v'=0$ 即 $\eta=e(g,g)^{a^{q+1}}$,否则 $v'=1$ 即 η 为群 G_1 中 1 个随机元素.

计算模拟器 B 得到正确猜测结果 $v'=v$ 的优势计算如下:

$$Adv_B=Pr[v'=v]-\frac{1}{2}=$$

$$Pr[v=0]Pr[v'=0|v=0]+$$

$$Pr[v=1]Pr[v'=1|v=1]-\frac{1}{2}=$$

$$\frac{1}{2}Pr[b'=b|v=0]+\frac{1}{2}Pr[v'=v|v=1]-\frac{1}{2}. \quad (51)$$

若 $Adv_A=\varepsilon$ 为不可忽略的优势,$v=0$ 时攻击者 A 在该游戏中获胜的概率为

$$Pr[b'=b|v=0]=Pr[v'=0|v=0]=\frac{1}{2}+\varepsilon. \quad (52)$$

$v=1$ 时,攻击者获胜的概率为

$$Pr[b'=b|v=1]=Pr[v'=1|v=1]=\frac{1}{2}. \quad (53)$$

于是 $Adv_B=\varepsilon/2$ 也为不可忽略的优势.模拟器可在多项式时间内以不可忽略的优势 $\varepsilon/2$ 解决 q-parallel BDHE 假设.

4　性能分析

本文通过 ABE 构建安全灵活的数据访问控制机制,并以此建立云存储服务体系,为用户提供安全可靠的数据存储和访问服务.同时,本文在该体系中引入高效安全的动态授权方法,增强了系统中数据的易管理性并提升了系统的运行效率.该动态授权方法能有效降低由于更新访问策略引起的计算及通信方面的资源耗费,表 1 对现有的 4 个方案中的访问策略更新方法做出了分析比较.

Table 1　Performance Comparison of Policy Updating Schemes

表 1　访问策略更新方案性能比较

Scheme	Form of Access Policy	Optimization of SSS Updating	Limit of Updating
Goyal et al's[1]	Access Tree	No	Yes
Sahai et al's[3]	LSSS Matrix	No	Yes
Yang et al's[4]	Different Updating Methods for Different Forms	No	No
Our Scheme	Universal Updating Methods for Different Forms	Yes	No

首先,由于本文通过算法 1 实现访问策略形式的无差别转换(保持逻辑结构及 shares 数值集合不变),使得本文的策略更新方法具备通用性而不依赖于具体策略形式,文献[1]和文献[3]分别针对访问树和 LSSS 矩阵提出其策略更新方法;而文献[4]中的方法虽然也支持多种形式的访问策略,但仅是针对每种具体的形式提出相应的更新方法法,通用性仍存在一定不足.其次,本文对上层的 SSS 方案的更新进行了优化:根据分块矩阵的特征,可以每个分块为单位进行对访问策略的更新、生成增量集合,其他文献中所提方案则需要针对策略中各点逐一计算其更新增量.特别地,文献[4]中提出的方法在对门限结构进行更新时,处理过程较繁琐、处理效率也较低.最后,本文的动态授权方法在对密文进行更新时没有任何限制,可进行任意的更新,其代价是需在本地存储部分密文信息,占用少量存储空间,而在文献[1]和文献[3]中,访问策略的更新是受到一定限制的.

以下分别对更新授权、代理授权和临时授权 3 种动态授权方式进行性能分析和比较.

4.1　更新授权性能分析

更新授权由数据所有者执行,本文定义了 4 个更新访问策略的基本函数:$Update_{\mathrm{Th}}$,$Update_{\mathrm{path}}$,$Remove_{\mathrm{subtree}}$,$Add_{\mathrm{subtree}}$.访问策略的任意更新均可经由这 4 个基本函数的组合操作实现.表 2 给出了这 4 个函数与计算及通信效率相关的评测参数:这 4 个算法对 LSSS 矩阵的更改较少、生成的增量集合 Δ 较小,因此数据所有者只需花费少量的计算代价和通信量代价即可实现访问策略的更新.更新策略的计算复杂度为 $O(|\Delta|)$,通信量为 $|\Delta|\times|g|+P$.其中,$|\Delta|$ 表示集合 Δ 中元素的个数,$|g|$ 表示双线性群 G_0 中 1 个元素所占用的字节数,P 表示更新矩阵所需传输的参数大小.由于仅需给定几个相关的参数,即可由云存储服务器自行更新 LSSS 矩阵,因此 P 远小于密文增量集合的大小.

为进一步对该动态授权方法进行评测,本文对所提出的方案进行了仿真实验.本文所有实验均在 1 个 Linux 虚拟机(CPU 2.78 Hz, RAM 1 GB)下执行,实验中通过 pbc lab(pairing based encryption lab)

Table 2 Evaluation Parameters of Policy Updating Schemes

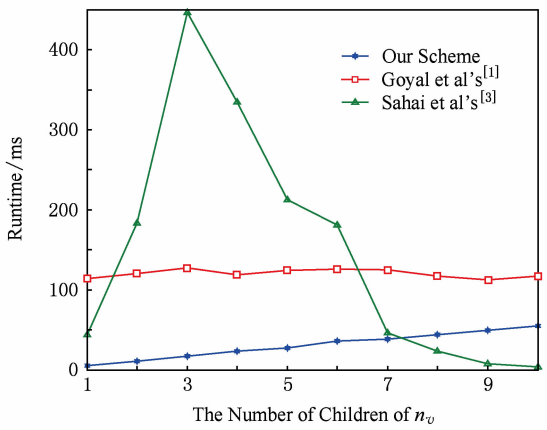
表 2 更新授权中主要函数性能评测参数

Function		Modifying of LSSS Matrix		$\|\Delta\|$	Updated $C_{i,j}$
		Updated Rows/Columns	Parameters for LSSS Matrix Updating		
$Update_{\mathrm{Th}}(n_v, t_v)$	$t_v > t$	Add Δt columns	$\Delta t, node_v$	n	l
	$t_v < t$	remove Δt columns	$\Delta t, node_v$	n	l
$Update_{\mathrm{path}}(r_v, p_v)$		Keep Size	r_v, p_v	1	l
$Remove_{\mathrm{subtree}}(r_v)$		Delete l rows, m columns	r_v	0	0
$Add_{\mathrm{subtree}}(r_v, p_v)$		Add l rows, m columns	$p_v, \boldsymbol{M}(r_v)$	l	l

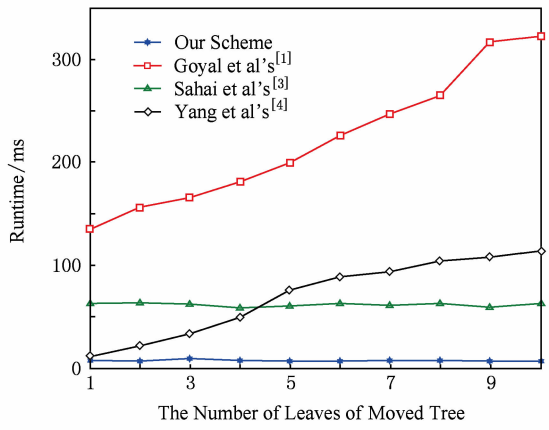
Note: n_v denotes a t-out-of-n node; t_v denotes the threshold of n_v; $\Delta t = |t - t_v|$; l denotes the number of leaves of the subtree with root n_v/r_v; m denotes the number of columns of submatrix which describes the subtree with root n_v/r_v.

包中提供的 a 型曲线生成双线性群 G_0，该群的阶 $|G_0|$ 为 160 b 的质数，群中元素 g 的大小 $|g|=512$ b. 实验结果如图 6 所示，图 6 中所统计的数据均为程序执行 20 次所测数据的平均值. 由于文献[2]中该操作的运行时间及通信效率与该文中定义的 lr 集合中节点个数成正比，其浮动较大，本文仅考虑所需操作子树中所关联的属性均连续排列的情况. 在该条件下，文献[2]中的策略更新所需平均运行时间及通信代价较低，且分析较为简单.

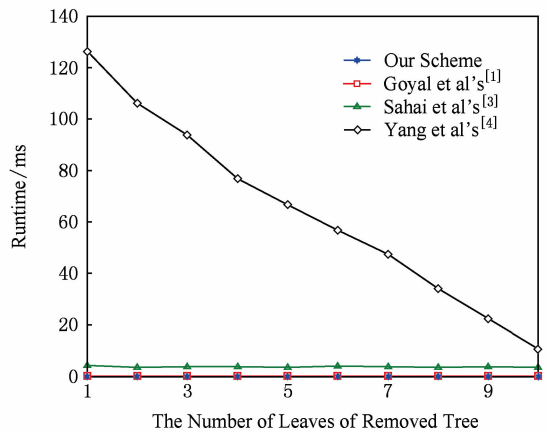
设 n_v 为访问策略中的 1 个中间节点，n_v 下的子树的叶子节点总数记为 l. 图 6(a)和图 6(e)所示为 $l=10$ 时各方案中数据所有者更改 n_v 的阈值 t 所需运行时间及通信量. 在本文提出的方法中，更改阈值 t 的运行时间及通信与 n_v 的子节点个数 n 成正比，而文献[1]中该操作的执行时间及通信量均与 n_v 下叶子节点总数 l 成正比. 文献[2]中所提供方法运行效率及通信量均与 lr 集合中节点个数成正比，其浮动较大. 显然，本文方法具备较高的执行效率及较低的通信代价. 文献[3]对门限结构的处理较为复杂，需要将该节点转换为或门和与门的组合，再转换为



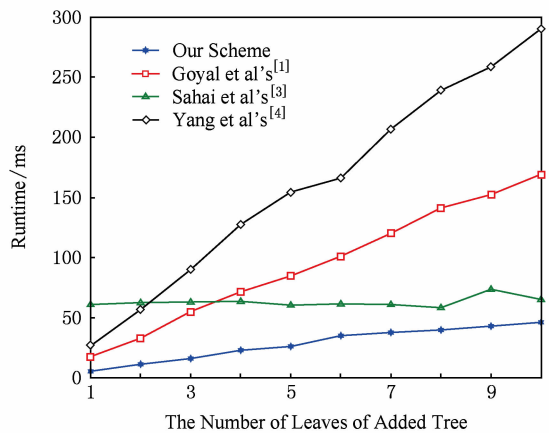
(a) Runtime of $Update_{\mathrm{Th}}$



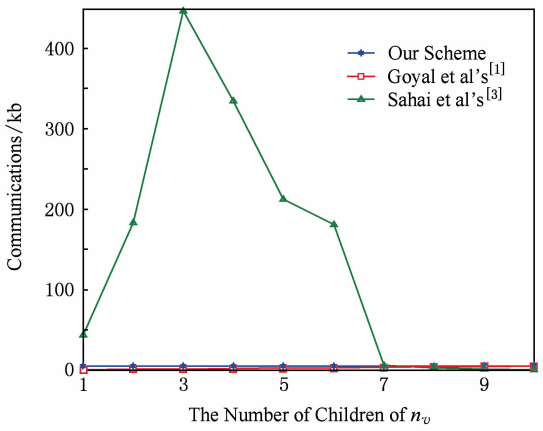
(b) Runtime of $Update_{\mathrm{path}}$



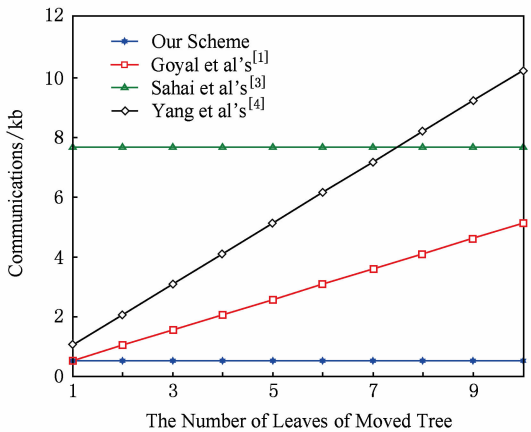
(c) Runtime of $Remove_{\mathrm{subtree}}$



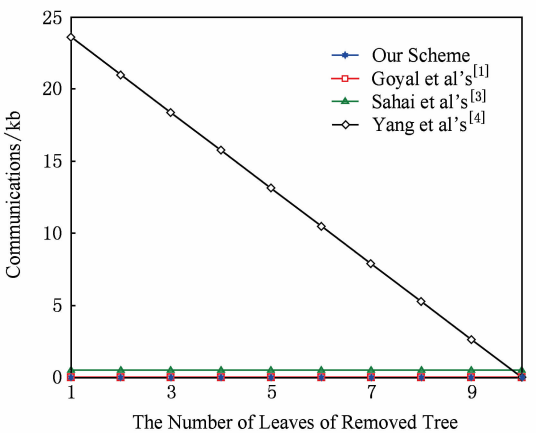
(d) Runtime of Add_{subtree}



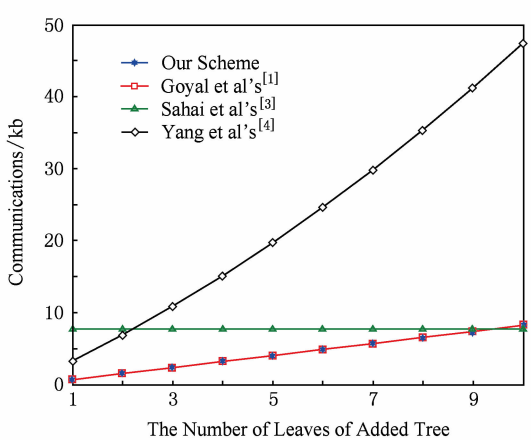
(e) Communications of $Update_{\mathrm{Th}}$



(f) Communications of $Update_{\mathrm{path}}$



(g) Communications of $Remove_{\mathrm{subtree}}$



(h) Communications of Add_{subtree}

Fig. 6　Runtime and communications of updating privilege.

图 6　更新授权操作的运行时间及通信量

相应的 LSSS 矩阵,效率明显低于其他方案,不宜一起做比较. 图 6(b)及图 6(f)所示为数据所有者移动 1 棵子树的位置时所需的运行时间及通信量. 文献[1,3-4]中没有专门针对此类操作的算法,该操作只能分为 2 步执行:先删除对应子树,再在所指定的位置插入相同结构的子树. 由图 6 可知本方案中所提方法的执行效率明显高于其他方案而通信代价也相对较小. 图 6(c)及图 6(g)所示为 $l=10$ 时数据所有者删除 n_v 下的 1 棵子树所需的操作时间和通信量. 在本文及文献[1]提出的方法中,只需指定待删除叶子节点对应的密文 $C_{i,j}$ 即可. 文献[3]中要先计算 lr,再根据 lr 集合计算更新节点密文,其计算量及通信量均与 lr 中元素个数成正比. 文献[4]中,在删除节点时存在 2 种情况:若 n_v 为或门,该方案与其他方案的操作类似,直接进行删除操作即可;若 n_v 为与门,文献[4]中需重构 SSS 方案,其操作时间与 n_v 下剩余的叶子节点数成正比. 显然,图 6(c)中黑色曲线统计的是文献[4]删除 AND 节点下的叶子节点所花费的时间. 图 6(d)和图 6(h)所示为 $l=10$ 时数据所有者在 n_v 下插入 1 棵子树时所需的操作时间及通信量,如图 6 所示,本方案的执行时间明显少于其他方案而通信代价也相对较小.

由实验结果可知,本文所提更新授权方法通过调用 $Update_{\mathrm{Th}}$,$Update_{\mathrm{path}}$,$Remove_{\mathrm{subtree}}$,$Add_{\mathrm{subtree}}$ 四个函数实现访问策略的更新,其执行效率明显高于其他方案,同时其通信代价也大幅度降低.

4.2　代理授权性能分析

代理授权由数据所有者指定的第三方代理执行,数据所有者仅需在生成密文的同时指定 1 个代理节点并生成 1 个代理集合 S_a 交付给第三方代理即可,代理节点可根据代理集合对代理子树进行管理,实现间接授权. 对于需要频繁更改访问策略的数据,代理授权有效降低了数据所有者的计算、通信开销并提升整个属性加密机制的性能. 图 7 为采用代理授权与直接进行重加密的实现效果对比图.

设有 LSSS 矩阵 $\boldsymbol{M}$,矩阵行数为 10. 图 7(a)及

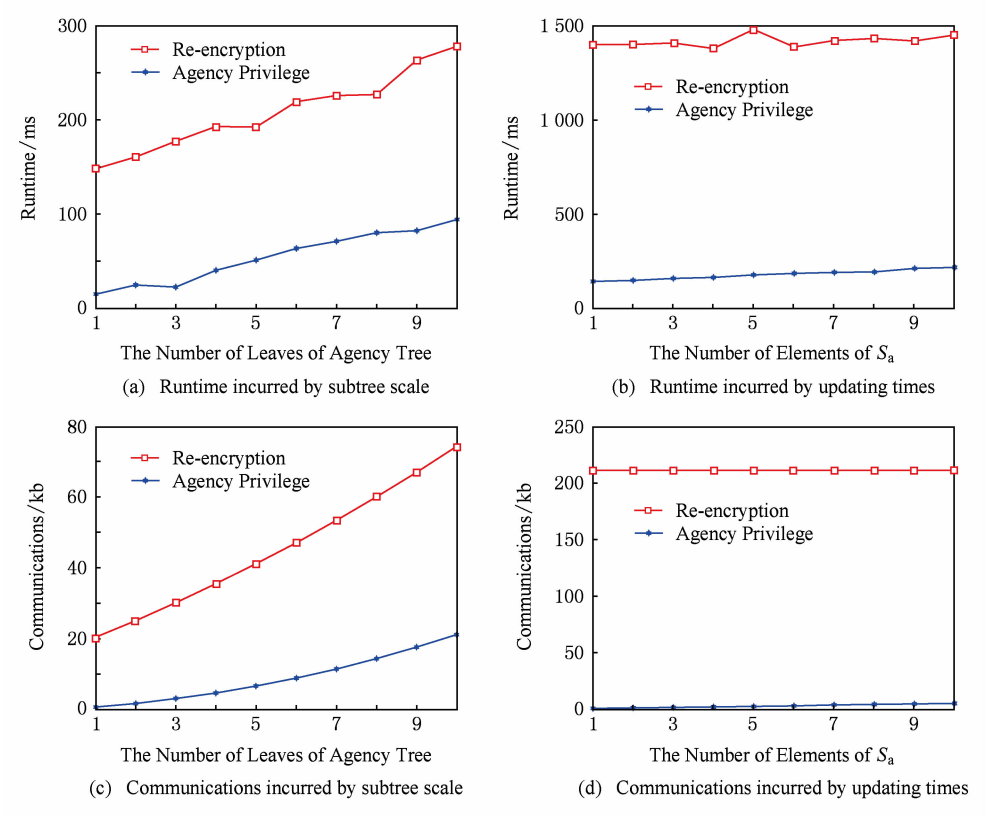


Fig. 7 Runtime and communications of agency privilege and re-encryption.

图 7 代理授权与重加密的运行时间及通信量

图 7(c)所示为在访问策略中插入 1 棵代理子树和执行同等效果的重加密所需的执行时间及通信量. 其中,委托代理三方插入代理子矩阵(即初始化代理子树)及相应密文时所花费时间与代理子树中的叶子节点数成正比. 与之相比,执行重加密所花费的时间远远大于通过代理授权实现策略更新所需花费的时间. 图 7(c)中曲线表示相应操作所需通信量,代理加密的通信代价明显低于重加密. 此后,代理方也可采用更新授权中的 4 个更新函数来提高代理授权的执行效率、降低通信量,其效果与更新授权类似.

图 7(b)和图 7(d)所示为连续更新访问策略 10 次时,数据所有者采用代理授权和执行重加密分别所花费的时间及通信量,图 7(b)(d)中横坐标表示集合 S_a 的元素的个数. 访问策略更新越频繁,数据所有者执行重加密时所花费的时间也越多,通信代价也越大;而采用代理授权,数据所有者只需花费计算 S_a 的时间即可实现多次的间接授权,相应的通信量也只与集合 S_a 中元素的个数相关.

4.3 临时授权性能分析

本文所提出的临时授权方法可将临时密文的访

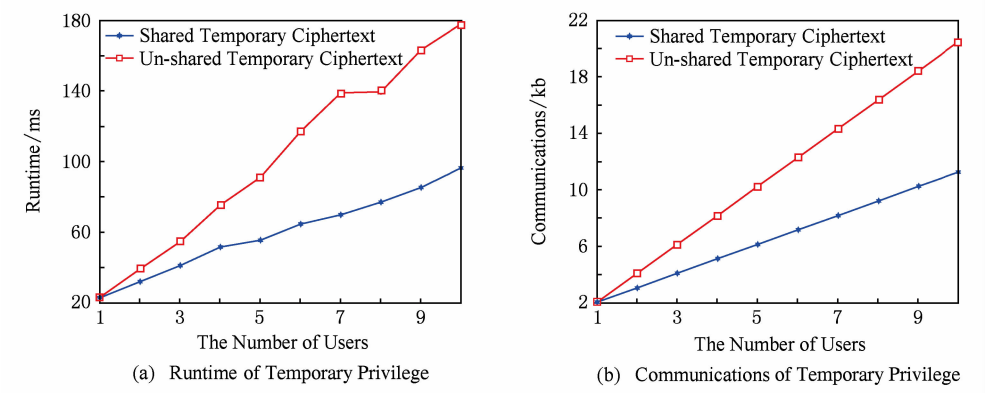


Fig. 8 Runtime and communications of temporary privilege.

图 8 临时授权的运行时间及通信量

问权限同时授予多个用户,且分别为各个用户生成会话密钥,该方法可极大地提升临时授权的运行效率.图 8(a)及图 8(b)中所示为 *CA* 将临时访问权限同时授予多个用户时所花费的时间及通信量,其中一条曲线表示为多个用户分别生成临时密文及会话密钥的情况,另一条曲线表示多个用户共享 1 个临时密文的情况.图 8 中横坐标表示被授予临时访问权限的用户个数,随着用户个数的增加,共享临时密文时计算量及通信量的增长并不大,对 *CA* 造成的负载相对较小.

5　结束语

云存储通过网络对分布式存储资源进行整合利用,在云存储体系中数据的安全性及易管理性是评测系统性能的 2 项重要指标.本文首先借助 ABE 机制设计出一种安全可靠的云存储服务体系,该体系中数据所有者通过访问策略自主定义数据的访问权限,并将数据以密文形式上传至云端,以确保云端数据的安全性.其次,本文提出了常见的访问策略之间的转换方法,通过该方法转换的访问策略在逻辑上和数值上均保持不变.最后,本文在此基础上提出了一种通用的访问策略动态更新方法,实现了支持动态授权的属性加密机制.本文所提出的动态授权方法根据授权方的不同分为 3 类:数据所有者执行的更新授权、第三方代理执行的代理授权以及 CA 执行的临时授权,并针对 SSS 方案的更改进行优化,降低了策略更新代价.这 3 类动态授权方式分别适用于不同的应用场景,能有效降低数据传输量、计算量,提升系统性能及执行效率.动态授权方法的实现使得数据所有者对加密数据的管理更为灵活便捷,使得属性加密机制能适用到更为广泛的应用环境中.

参　考　文　献

[1] Goyal V, Pandey O, Sahai A, et al. Attribute-based encryption for fine-grained access control of encrypted data [C] //Proc of the 13th ACM Conf on Computer and Communications Security. New York: ACM, 2006: 89-98

[2] Su Wenchi, Chang Shuchih Ernest. Integrated cloud storage architecture for enhancing service reliability, availability and scalability [C] //Proc of IEEE Int Conf on Information Science, Electronics and Electrical Engineering. Piscataway, NJ: IEEE, 2014: 764-768

[3] Sahai A, Seyalioglu H, Waters B, et al. Dynamic credentials and ciphertext delegation for attribute-based encryption [C] //Proc of the 32nd Cryptology Conf. Berlin: Springer, 2012: 199-217

[4] Yang Kan, Jia Xiaohua, Ren Kui, et al. Enabling efficient access control with dynamic policy updating for big data in the cloud [C] //Proc of the 33rd Annual IEEE Int Conf on Computer Communications. New York: IEEE Communications Society, 2014: 2013-2021

[5] Liu Zhen, Cao Zhenfu. On efficiently transferring the linear secret-sharing scheme matrix in ciphertext-policy attribute-based encryption[R/OL]. IACR Cryptology ePrint Archive, 2010 [2010-06-29]. http://eprint.iacr.org/2010/374

[6] Fu Yingxun, Luo Shengmei, Shu Jiwu. Secure online storage system based on cloud storage environment [J]. Journal of Software, 2014, 25(8): 1831-1843 (in Chinese)
(傅颖勋, 罗圣美, 舒继武. 一种云存储环境下的安全网盘系统[J]. 软件学报, 2014, 25(8): 1831-1843)

[7] Zhang Guigang, Li Chao, Zhang Yong, et al. A kind of cloud storage model research based on massive information processing [J]. Journal of Computer Research and Development, 2012, 49(Suppl 1): 32-36 (in Chinese)
(张桂刚, 李超, 张勇, 等. 一种基于海量信息处理的云存储模型研究[J]. 计算机研究与发展, 2012, 49(增刊 1): 32-36

[8] Buyya R, Chee S Y, Venugopal S. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities [C] //Proc of the 10th IEEE Int Conf on High Performance Computing and Communications. Piscataway, NJ: IEEE, 2008: 5-13

[9] Beimel A. Secure schemes for secret sharing and key distribution [D]. Haifa, Israel: Faculty of Computer Science, Technion-Israel Institute of Technology, 1996

[10] Andersen A, Trygve A, Schirmer N. Privacy for cloud storage [C] //Proc of Securing Electronic Business. Berlin: Springer, 2014: 211-219

[11] Ullah S, Zheng Xuefeng. TCLOUD: A trusted storage architecture for cloud computing [J]. International Journal of Advanced Science and Technology, 2014, 63(6): 65-72

[12] Su Jinshu, Cao Dan, Wang Xiaofeng, et al. Attribute based encryption schemes [J]. Journal of Software, 2011, 22(6): 1299-1315 (in Chinese)
(苏金树, 曹丹, 王小峰, 等. 属性基加密机制[J]. 软件学报, 2011, 22(6): 1299-1315)

[13] Sahai A, Waters B. Fuzzy identity-based encryption [C] // Proc of the 24th Annual Int Conf on The Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2005: 457-473

[14] Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption [C] //Proc of the IEEE Symp on Security and Privacy. Los Alamitos, CA: IEEE Computer Society, 2007: 321-334

[15] Goyal V, Jain A, Pandey O, et al. Bounded ciphertext policy attribute based encryption [C] //Proc of the 35th Int Colloquium on Automata, Languages and Programming. Berlin: Springer, 2008: 579-591

[16] Xhafa F, Li Jingwei, Zhao Gansen, et al. Designing cloud-based electronic health record system with attribute-based encryption [J/OL]. Multimedia Tools and Applications, 2014 [2015-02-15]. http://link.springer.com/article/10.1007/s11042-013-1829-6

[17] Li Jin, Chen Xiaofeng, Li Jingwei, et al. Fine-grained access control system based on outsourced attribute-based encryption [C] //Proc of the 18th European Symp on Research in Computer Security. Berlin: Springer, 2013: 592-609

[18] Waters B. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization [C] //Proc of the 14th Int Conf on Practice and Theory in Public Key Cryptography. Berlin: Springer, 2011: 53-70

[19] Yang Kan, Jia Xiaohua, Ren Kui, et al. Dac-macs: Effective data access control for multi-authority cloud storage systems [J]. IEEE Trans on Information Forensics and Security, 2013, 8(11): 1790-1801

[20] Hohenbberger S, Waters B. Attribute-based encryption with fast decryption [C] //Proc of the 16th Int Conf on Practice and Theory in Public-Key Cryptography. Berlin: Springer, 2013: 162-179

[21] Naruse T, Mohri M, Shiraishi Y. Attribute-based encryption with attribute revocation and grant function using proxy re-encryption and attribute key for updating [C] //Proc of the 5th Int Conf on Future Information Technology. Berlin: Springer, 2014: 119-125

[22] Lai Junzuo, Deng Robert, Yang Yanjiang, et al. Adaptable ciphertext-policy attribute-based encryption [C] //Proc of the 6th Int Conf on Pairing-Based Cryptography. Berlin: Springer, 2014: 199-214

[23] Waters B. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization [C] //Proc of the 14th Int Conf on Practice and Theory in Public Key Cryptography. Berlin: Springer, 2011: 53-70

[24] Song Yun, Li Zhihui, Li Yongming, et al. A new multi-use multi-secret sharing scheme based on the duals of minimal linear codes [J]. Security and Communication Networks, 2014, 8(2): 202-211

[25] Rao Y S, Dutta R. Dynamic ciphertext-policy attribute-based encryption for expressive access policy [C] //Proc of the 10th Int Conf on Distributed Computing and Internet Technology. Berlin: Springer, 2014: 275-286

[26] Hohenberger S, Waters B. Online/offline attribute-based encryption [C] //Proc of the 17th Int Conf on Practice and Theory in Public-Key Cryptography. Berlin: Springer, 2014: 293-310

Wang Jing, born in 1986. PhD of Wuhan University. Her research interests include security and privacy of cloud (wjing@whu.edu.cn).

Huang Chuanhe, born in 1963. Professor and PhD supervisor of Wuhan University. Member of China Computer Federation. His main research interests include cryptography, wireless security and trust management, SDN, opportunistic network and wireless network.

Wang Jinhai, born in 1982. PhD of Wuhan University. His research interests include cloud computing and high performance computing.