软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2020,31(12):3923−3936 [doi: 10.13328/j.cnki.jos.005856]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

一种支持快速加密的基于属性加密方案*

罗王平[1], 冯朝胜[1,2], 邹莉萍[1], 袁 丁[1], 吴唐美[1], 李 敏[1], 王广杰[3]

[1](四川师范大学 计算机科学学院,四川 成都 610101)

[2](可视化计算与虚拟现实四川省重点实验室(四川师范大学),四川 成都 610101)

[3](四川师大科技园发展有限公司,四川 成都 610066)

通讯作者: 冯朝胜, E-mail: csfenggy@126.com

摘 要: 基于属性加密算法因含有大量耗时的指数运算和双线性对运算,一些方案提出将加密外包给云服务器.然而这些方案并没有给出外包加密在云服务器中的并行计算方法,而且还存在用户保管私钥过多、授权中心生成用户私钥成本过大的问题.针对这些问题,提出一种基于 Spark 大数据平台的快速加密与共享方案.在该方案中,根据共享访问树的特点设计加密并行化算法,该算法将共享访问树的秘密值分发和叶子节点加密并行化之后交给 Spark 集群处理,而用户客户端对每个叶子节点仅需要一次指数运算;此外,用户私钥的属性计算也外包给 Spark 集群,授权中心生成一个用户私钥仅需要 4 次指数运算,并且用户仅需要保存一个占用空间很小的密钥子项.

关键词: 基于属性加密;加密外包;快速加密;Spark 平台

中图法分类号: TP309

中文引用格式: 罗王平,冯朝胜,邹莉萍,袁丁,吴唐美,李敏,王广杰.一种支持快速加密的基于属性加密方案.软件学报,2020,31(12):3923−3936. http://www.jos.org.cn/1000-9825/5856.htm

英文引用格式: Luo WP, Feng CS, Zou LP, Yuan D, Wu TM, Li M, Wang GJ. Attribute-based encryption scheme with fast encryption. Ruan Jian Xue Bao/Journal of Software, 2020,31(12):3923−3936 (in Chinese). http://www.jos.org.cn/1000-9825/5856.htm

Attribute-based Encryption Scheme with Fast Encryption

LUO Wang-Ping[1], FENG Chao-Sheng[1,2], ZOU Li-Ping[1], YUAN Ding[1], WU Tang-Mei[1], LI Min[1], WANG Guang-Jie[3]

[1](School of Computer Science, Sichuan Normal University, Chengdu 610101, China)

[2](Visual Computing & Virtual Reality Key Laboratory of Sichuan Province (Sichuan Normal University), Chengdu 610101, China)

[3](Sichuan Normal University Technology Park Development Co., Ltd, Chengdu 610066, China)

Abstract: Attribute-based encryption algorithm contains a large number of time-consuming exponential operations and bilinear pairing operations, therefore, some schemes propose to outsource encryption to the cloud server. However, these schemes do not provide the parallel computing method of outsourcing encryption on cloud servers. Besides, in these schemes, user manages too many private keys and the authorization center generates a private key for the user with excessive cost. To solve these problems, a fast encryption and sharing scheme based on the Spark big data platform is proposed. In this scheme, an encryption parallelization algorithm is designed according to the characteristics of the sharing access tree, with which, distribution of secret value of the sharing access tree and encryption at leaf node are parallelized. Then, the parallelization tasks are handed over to the Spark cluster. As a result, user client needs only one

* 基金项目: 国家自然科学基金(61373163); 国家科技支撑计划(2014BAH11F02); 四川省科技支撑计划(2015GZ079); 四川师范大学研究生优秀论文培育基金(川师研[2018]3 号-38); 国防科技重点实验室项目(6142103010709)

Foundation item: National Natural Science Foundation of China (61373163); National Key Technology Research and Development Program of theMinistry of Science and Technology of China (2014BAH11F02); Science and Technology Support Program of Sichuan Province (2015GZ079); Postgraduate Excellent Paper Cultivation Fund of Sichuan Normal University (Chuan Shi Yan [2018] No.3-38); Project of Key Laboratory of National Defense Science and Technology (6142103010709)

收稿时间: 2018-04-02; 修改时间: 2018-08-30; 采用时间: 2019-04-25

exponent operation for each leaf node. In addition to this, the private key attribute computation is also outsourced to the Spark cluster. In proposed scheme, the authorization center generates a user private key requiring only four exponential and users only need to save a key sub-item with small space.

Key words: attribute-based encryption; encryption outsourcing; fast encryption; Spark platform

在诸如 GFS[1]的外包存储环境中,基于属性加密 ABE(attribute-based encryption)[2]因能实现数据的安全存储、访问控制和秘密共享,引起了学术界的广泛关注[3−6].根据访问控制策略与用户私钥相关还是与数据密文相关,ABE 可以分为密钥策略基于属性加密 KP-ABE(key-policy attribute-based encryption)[7]和密文策略基于属性加密 CP-ABE(ciphertext-policy attribute-based encryption)[8].但是 ABE 的数学基础之一是双线性映射[9,10],而双线性映射存在耗时的指数运算和双线性对运算,导致 ABE 算法的计算效率低下,算法的计算要求,对于包括智能手机、平板在内的电子设备都很难满足.为了提高 ABE 算法的计算效率,减轻用户客户端的计算负担,一些引入计算外包的 ABE 方案被提出.然而这些方案仅仅研究了如何将用户客户端的大部分计算工作外包给云服务器,没有给出外包加密在云服务器中的并行计算方法,也没有解决外包加密的数据安全问题,而且还存在用户保管私钥过多、授权中心生成用户私钥成本过大的问题.针对上述问题,在 Green 等人所提出的 CP-ABE 方案[11]基础上,利用 Spark 技术的 Map 操作和 Reduce 操作,提出一种面向公有云的快速加密与共享方案.该方案的具体贡献如下.

(1) 提出一种基于 Spark 大数据平台的快速加密与共享方案.在该方案中,将密文共享访问树的计算工作外包给云服务器,而用户客户端仅需要执行 4 次指数运算以及每个叶子节点的一次指数运算;

(2) 设计一种快速计算密文共享访问树的方法.该方法先将共享访问树的计算任务分解成多个可并行执行的小任务,然后将这些小任务交给 Spark 集群处理.使用该并行计算方法,共享访问树的计算效率得到显著提高;

(3) 提出一种用户私钥生成外包方法.该方法将用户属性对应的指数运算外包给 Spark 集群,而授权中心仅需要 4 次指数运算就能生成一个用户私钥,并且将大多数密钥子项交由云服务器保存,用户客户端仅需要保存一个占用空间很小的密钥子项.

本文第 1 节对 CP-ABE 算法的研究现状进行总结,并分析它们的优缺点.第 2 节概述本文提出的一种面向公有云的快速加密与共享框架.第 3 节给出本文方案的安全模型.第 4 节详细介绍本文提出的一种面向公有云的快速加密与共享方案.第 5 节从安全性和性能上对本文方案进行分析.第 6 节对本文方案进行实验分析.最后对全文进行总结.

1 相关研究

2007 年,为了保证外包数据的安全存储和秘密共享,Bethencourt 等人[8]率先提出来密文策略基于属性加密方案.该方案的用户私钥与属性集合相关联,数据密文与访问控制策略相关联,其不但接近基于角色的访问控制方法,而且能抵抗用户间的共谋攻击.然而,其仅仅在随机预言模型下能达到选择明文安全.为了解决这一问题,Waters 采用线性秘密共享方案 LSSS(linear secret sharing scheme)[12,13],提出一种在标准模型下能达到选择明文安全的 CP-ABE 方案[14].该方案高效、可靠且更具表达力,并给出了安全性证明,但其加解密时间随访问表达式的复杂度呈线性增长.2011 年,Green 等人[11]在 Waters 的基础上提出一种具有可重放选择密文安全的外包解密方案,该方案选择一个随机因子将用户私钥生成转换密钥,并提交给云服务器进行部分解密,然后将云服务器部分解密结果下载至用户客户端,利用该随机因子解密出最后的明文数据,大大减轻了用户客户端的计算负担.该方案同时给出了 CP-ABE 和 KP-ABE 两种方案的外包解密方法,但没有给出外包加密方法.为此,Zhou 等人[15]于 2012 年提出一种外包加密方法,该方法将共享访问树绝大部分计算工作外包给云服务器完成,而用户客户端仅需要计算共享访问树的根节点和一个叶子节点.该方案尽管在外包加密的同时云服务器不能获取数据明文,但无法抵抗满足未外包叶子节点属性的用户与云服务器之间的共谋攻击.同年,Li 等人[16]受 Zhou 的启发,提出一种基于 MapReduce 技术的外包加密方案.该方案将共享访问树从根节点分为两部分:根节点右边为一个叶子

节点且由用户客户端完成计算,而其左子树的计算工作则交由 MapReduce 完成.首先,将根节点分发给左孩子节点的秘密值分成N份,针对每一份秘密值,启动一个 Map 完成左子树秘密值分发和叶子节点密文计算,而 Reduce 针对左子树每个叶子节点,将 Map 的输出结果连乘得到最后完整的加密结果.然而该方案基于 MapReduce 集群的主节点和至少一个数据节点是可信的,在实际应用环境中难以实施.2013 年,Lai 等人[17]提出一种可验证的外包解密方案,该方案首先选择一个随机因子,将其和数据明文采用哈希运算和幂运算生成一个用于验证的密文子项,对随机因子和数据明文同时用基于属性加密算法加密.解密时,用与 Green 类似的外包解密方案分别解密出随机因子和数据明文,重新生成用于验证的密文子项并与加密时生成的密文子项做一致性对比,以达到可验证的目的.但该方案同时对随机因子和数据明文进行基于属性加密,其用户客户端的加密计算量与一般方案相比增长了一倍.2015 年,Qin 等人[18]提出的一个高效率且可验证的外包解密方案则较好地解决了这一问题,该方案与 Lai 的方案一样,首先选择一个随机因子,并对其进行基于属性加密;但对于数据明文,仅使用随机因子对其进行对称加密,并采用哈希方式对外包解密进行正确性验证.同年,Lin 等人[19]也提出一种可验证的外包解密方案.与其他方案不同的是,该方案生成所有密文子项但数据明文不嵌入其中,并将随机因子和数据明文级联后与访问控制策略解密后的秘密值做异或运算.与 Lai 的方案相比,该方案将通信带宽和计算成本降低了近一半.2017 年,Huang 等人[20]将具有强大计算能力的雾计算引入到 CP-ABE 方案的设计之中,提出一种外包加密的访问控制方案.该方案首先选择一个随机因子,将随机因子外包给雾计算进行基于属性加密,而用户客户端利用随机因子作为对称加密密钥加密数据明文,并对雾计算的计算结果做进一步处理,得到完整的密文数据.该方案使得大部分计算工作得到外包,而用户客户端仅需要完成少量的计算工作.但该方案将共享访问树的秘密值完全暴露给雾计算节点,使得其存在安全问题.

上述分析表明:已有一些 CP-ABE 方案通过外包加密来解决 ABE 加密慢、效率低的问题,但这些方案只是简单地将部分加密工作外包给云计算这样的外包环境,并没有利用并行化计算方法来提升加密速度和效率.另外,基于属性加密主要利用树和矩阵两种结构来表示访问策略和实现秘密共享;而和访问矩阵相比,访问树具有更容易构造、可读性更强、加密效率更高等优点.鉴于以上原因,该文对利用树表示访问策略的 CP-ABE 方案的外包加密的并行化开展研究.

2 面向公有云的快速加密与共享框架

2.1 基于Spark的快速加密方法

计算秘密共享数是 CP-ABE 实现密文共享十分重要的一个环节.已有的基于访问结构树的 ABE 方案,计算秘密数基本都是从根节点到叶子节点逐层依次按串行方式进行,难以快速计算出各叶子节点秘密共享数.通过观察发现(如图 1 所示),叶子节点的秘密共享数为从根节点到叶子节点路径上每个节点(根节点除外)在其父节点对应多项式(去掉常数项)的取值以及秘密数之和.基于以上观察,可以得出定理 1.

定理 1. 访问结构树各叶子节点的秘密共享数为从根节点到叶子节点路径上每个节点(根节点除外)在其父节点对应多项式(去掉常数项)的取值以及秘密数之和.

证明:设秘密数为 s,访问结构树从根节点到某叶子节点的路径为 $r_0 \to r_1 \to \ldots \to r_n, r_0$ 和 r_n 分别对应根节点和叶子节点;节点 $r_i (i \in [0,n])$对应的多项式为 $f_i(x)$,对应的无常数项多项式记作 $f_i'(x)$;函数 $index(x)$表示求节点 x 在兄弟节点中的序号.各节点的秘密共享数如下:

$$
\begin{aligned}
&f_0(0) = s,\\
&f_1(0) = f_0'(index(r_1)) + s,\\
&f_2(0) = f_1'(index(r_2)) + f_1(0) = f_1'(index(r_2)) + f_0'(index(r_1)) + s,\\
&\ldots,\\
&f_n(0) = f_{n-1}'(index(r_n)) + f_{n-1}(0) = f_{n-1}'(index(r_n)) + \ldots + f_1'(index(r_2)) + f_0'(index(r_1)) + s.
\end{aligned}
$$

所以,命题成立. □

定理 1 表明,各个叶子节点秘密共享数的计算可以按照并行方式进行.因此,设计一个基于 Spark 并行计算

模型,能快速求出秘密共享数和叶子节点密文的计算方案.在该方案中,针对秘密共享数,每个非叶子节点对应于一个 Map 操作,每个叶子节点对应于一个 Reduce 操作,并且每个 Map 操作需要知道对应节点其子树中所有的叶子节点(若考虑每个 Map 操作计算一个节点在启动 Map 时间与计算节点时间之比太高,可考虑单个 Map 操作计算多个节点,具体几个可根据实验结果进行);每个 Map 操作为每个节点选择一个多项式,但常数项为 0(根节点常数项为 s),然后计算 $Q(1),Q(2)$等,如果叶子节点在 Map 操作对应节点的第 1 个子树上,就发送 $Q(1)$的值给对应叶子节点的 Reduce 操作,如图 1 的节点 a,将 $Q(1)$发送给 A,B,C,$Q(2)$发送给 E,等等;发送内容为〈叶子节点 Reduce 标识,$Q(index)$〉,最后,叶子节点对应的 Reduce 将收到的值相加,即为秘密分量;针对叶子节点密文计算,每个叶子节点与随机数相关的指数运算对应于一个 Map 操作,将其发送给对应叶子节点的 Reduce 操作,Reduce 操作进行秘密分量的指数运算,从而得到叶子节点完整密文.详细流程如图 1 所示.

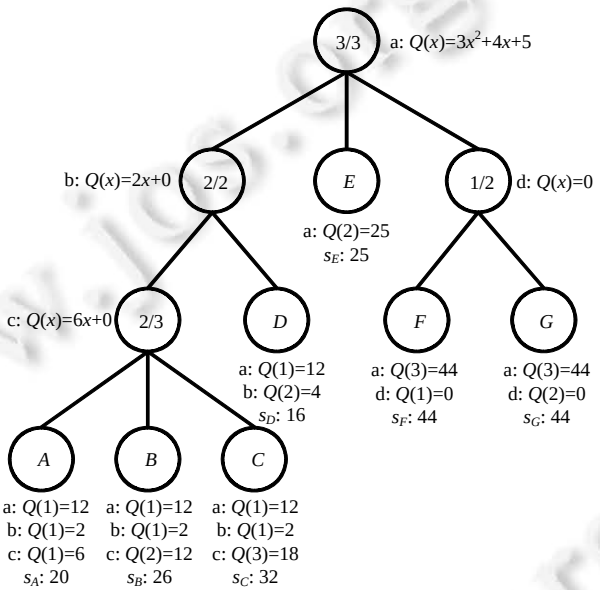

Fig.1 Flow chart of secret value parallel distribution

图 1 秘密值并行分发流程图

2.2 安全共享机制

计算外包的难点在于,实现外包的同时还需要保证数据的安全.外包时,将密文共享访问树的计算工作外包给 Spark 集群处理,但为了保证数据安全,Spark 集群计算共享访问树时,每个节点的多项式常数项均为 0,并在叶子节点获取秘密分量后加一,计算完成后,将与属性相关的密文子项发回用户客户端与秘密值 s 做幂运算.

定理 2. 共享访问树在秘密值为 0 且叶子节点秘密分量加一或秘密值为 1 的情况下,相同叶子节点获得的秘密分量相同.

证明:当共享访问树设置秘密值为 1 时,由定理 1 可知,叶子节点 r_n 的秘密分量为

$$f_{n,1}(0)=f'_{n-1}(index(r_n))+\ldots+f'_1(index(r_2))+f'_0(index(r_1))+1.$$

当共享访问树设置秘密值为 0 时,由定理 1 可知,叶子节点 r_n 的秘密分量为

$$f_{n,0}(0)=f'_{n-1}(index(r_n))+\ldots+f'_1(index(r_2))+f'_0(index(r_1))+0.$$

再加上一可得:

$$f_{n,2}(0)=f_{n,0}(0)+1=f'_{n-1}(index(r_n))+\ldots+f'_1(index(r_2))+f'_0(index(r_1))+1.$$

即 $f_{n,1}(0)=f_{n,2}(0)$,故命题成立. □

定理 3. 共享访问树在秘密值为 1 并且叶子节点秘密分量乘以 s 或秘密值为 s 的情况下,根节点的秘密值

相同.

证明:当共享访问树设置秘密值为 s 时,由定理 1 可知,叶子节点 r_n 的秘密分量为

$$f_{n,1}(0)=f'_{n-1}(index(r_n))+\ldots+f'_1(index(r_2))+f'_0(index(r_1))+s.$$

当共享访问树设置秘密值为 1 时,由定理 1 可知,叶子节点 r_n 的秘密分量为

$$f_{n,0}(0)=f'_{n-1}(index(r_n))+\ldots+f'_1(index(r_2))+f'_0(index(r_1))+1.$$

再乘以 s 可得:

$$f_{n,2}(0)=f_{n,0}(0)\cdot s=f'_{n-1}(index(r_n))\cdot s+\ldots+f'_1(index(r_2))\cdot s+f'_0(index(r_1))\cdot s+s.$$

由定理 1 可知,秘密分量 $f_n(0)$的最后一项为加密时根节点的秘密值,即两种情况在加密时根节点的秘密值均为 s,故命题成立. □

由定理 2 和定理 3 可知:共享访问树在秘密值为 0 且叶子节点秘密分量加一再乘以 s 或秘密值为 s 的情况下,根节点的秘密值相同.

2.3 面向公有云的快速加密与共享框架

在面向公有云的快速加密与共享框架中,主要包括授权中心、云服务器、数据所有者和数据消费者,如图 2 所示.授权中心位于用户可信域内,负责用户私钥的生成、更新和撤销;云服务器负责用户转换密钥和数据密文的存储和外包计算;数据所有者亦是数据消费者,能利用用户客户端计算部分密文子项,并将其发送至云服务器;数据消费者能从云端读取并解密授权共享密文.假设本文中的云服务器是“忠诚而好奇”的,即会按照用户的要求进行操作,但对用户的敏感数据产生好奇.

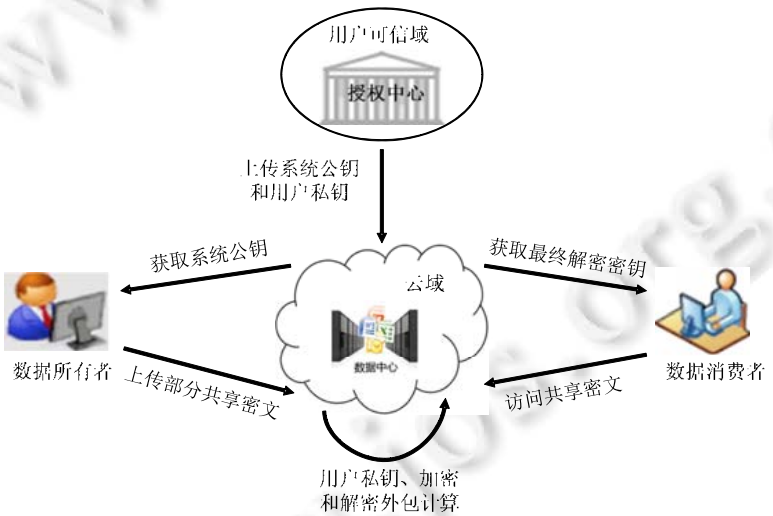

Fig.2 Fast encryption and sharing framework for public cloud

图 2 面向公有云的快速加密与共享框架

在该框架中,用户客户端首先计算除共享访问树以外的密文子项,将 g^s 和共享访问逻辑表达式发送至云服务器,云服务器利用定理 1 的并行计算方法完成共享访问树的计算工作,然后将与属性相关的密文子项发回用户客户端,并与秘密值 s 做幂运算,从而得到完整密文数据,将完整的密文数据发送至云服务器存储.

3 安全模型

下面给出定义支持快速加密的 CP-ABE 方案安全模型.

初始化:挑战者运行 Setup 算法,并将系统公钥 PK 发送给敌手.

阶段 1:挑战者初始化一个空表 T,一个空的集合 D 和一个整数 j=0.敌手可以重复进行以下任何查询.

- *Create*(*S*):挑战者设置 *j*:=*j*+1,利用外包私钥生成算法获取属性集 *S* 对应的(*SK*,*TK*),并将(*j*,*S*,*SK*,*TK*)存储在 *T* 中,然后将转换密钥 *TK* 发送给敌手.在相同输入的情况下,Create 能够被重复的查询;
- *Corrupt*(*i*):如果在表 *T* 中存在第 *i* 个记录,挑战者获取该记录(*i*,*S*,*SK*,*TK*),并设置 *D*:=*D*∪{*S*},然后将私钥 *SK* 返回给敌手;若不存在该记录,则返回⊥;
- *Decrypt*(*i*,*CT*):如果在表 *T* 中存在第 *i* 个记录,挑战者获取该记录(*i*,*S*,*SK*,*TK*),并将以(*SK*,*CT*)为输入的解密运算输出结果返回给敌手;若不存在该记录,则返回⊥.

挑战:敌手提交两个等长的明文 M_0 和 M_1;提交一个挑战访问树 T^*,使得任意 $S\in D$ 都不满足 T^*.挑战者抛硬币获得一个随机值 *b*,计算明文 M_b 关于访问树 T^*的密文 CT^*,并将 CT^*返回给敌手.

阶段 2:重复阶段 1 的查询,但不能进行如下查询.

- 获取能解密挑战密文的私钥;
- 解密挑战密文.

猜测:敌手给出一个关于 *b* 的猜测值 *b*′.

4 面向公有云的快速加密与共享方案

方案包括初始化、私钥生成、加密和解密这 4 个模块,利用 Spark 技术的 Map 操作和 Reduce 操作并行化计算用户私钥和数据密文.

(1) 初始化:*Setup*(*U*,*k*)

k 为系统安全参数,选择一个阶为大素数 *p* 的双线性群 *G* 和 G_T,从 *G* 中选择一个元素 *g* 并记为 *G* 的生成元,双线性映射 $e:G\times G\to G_T$,选择 $g_2\in G, a,\alpha\in Z_p^*$,对于系统中每个属性 $i\in U$,选择 $t_i\in Z_p^*$,计算 $Y=e(g,g_2)^{\alpha}$,系统属性 $T_i=g^{t_i}$.定义哈希函数 $H_1:\{0,1\}^*\to Z_p^*, H_2:\{0,1\}^*\to\{0,1\}^k$.将系统公钥 *PK* 向所有用户及云服务器公开,而系统主密钥 *MK* 由授权中心秘密保存.系统公钥 *PK* 如下:

$$PK=\langle p,g,g^a,G,G_T,e,Y,\{T_i\}_{i\in U},H_1,H_2\rangle.$$

系统主密钥 *MK*: $MK=\langle g_2^{\alpha},\{t_i\}_{i\in U}\rangle$.

(2) 生成用户私钥:*KeyGen*(*PK*,*MK*,*S*)

用户私钥先由用户完成初步计算,再外包给云服务器完成最后的计算.

首先,用户客户端选择随机数 $t,t_0,\beta\in Z_p^*$,生成用户部分转换密钥:

$$TK'=\langle S,K=(g^{at}g_2^{\alpha})^{\beta},L=g^{\beta t},T=g^{t_0},\{t_i\beta t/t_0\}_{i\in S}\rangle.$$

将 *TK*′发送至云服务器,再利用 Spark 技术的 Map 操作并行计算私钥子项 $K_i=T^{t_i\beta t/t_0}=T_i^{\beta t}$,完整的用户转换密钥 *TK* 如下:

$$TK=\langle S,K=(g^{at}g_2^{\alpha})^{\beta},L=g^{\beta t},\{K_i=T_i^{\beta t}\}_{i\in S}\rangle.$$

最终解密密钥 *DK*=⟨β⟩由用户秘密保存,而转换密钥 *TK* 由云服务器保存在转换密钥库中.

(3) 加密:*Encrypt*(*T*,*M*∈{0,1}^{k},*PK*)

加密计算过程由用户客户端和云服务器共同完成(如图 3 所示).首先,用户客户端随机选择 $R\in G_T$,计算 $s=H_1(R,M), r=H_2(R), C=R\cdot Y^s=R\cdot e(g,g_2)^{\alpha s}, C'=M\oplus r, C_0=g^s$,将 g^s 和共享访问逻辑表达式发送至云服务器.

云服务器根据共享访问逻辑表达式构建密文共享访问树 *T*,采用 Spark 技术的 Map 和 Reduce 两种操作完成访问树 *T* 的秘密值分发与叶子节点属性加密计算,其中:Map 操作并行计算访问树 *T* 的每个节点(包括叶子节点和非叶子节点),而 Reduce 操作针对访问树 *T* 的每个叶子节点作进一步计算.令 $I_{j,child}$ 为非叶子节点 I_j 的孩子节点,其包含的叶子节点 L_i 集合为 $\{L_i\}_{L_i\in I_{j,child}}$,则非叶子节点 I_j 包含的叶子节点集合为 $L_{set}=\{\{L_i\}_{L_i\in I_{j,child}}\}_{I_{j,child}\in I_j}$.

- Map 操作

该部分按非叶子节点 I_j 和叶子节点 L_i 分为如下两种情况.

➢ 针对非叶子节点 I_j,Map 操作输入的键值对为(I_j,L_{set}).首先,根据其阈值 *d*,随机选择一个一元 *d*–1 次多

项式 $Q_j(x)$,其中,$0=Q_j(0)$且 $Q_j(x)$的其他系数从群 Z_p^* 中随机选择.再对非叶子节点 I_j 的每个孩子节点 $I_{j,child}$ 计算部分秘密值 $s_{i,j}=Q_j(index(I_{j,child}))$(其中,$index(I_{j,child})$表示孩子节点 $I_{j,child}$ 在兄弟节点中的序号,从左往右进行编号),将该秘密值输出至每个叶子节点 $L_i \in I_{j,child}$ 所在 Reduce 操作,即 Map 操作输出的键值对为$(L_i,s_{i,j})$;

- 针对叶子节点 L_i,Map 操作输入的键值对为(L_i,g^s),随机选择 $r_i' \in Z_p^*$,令 $r_i = r_i' \cdot s$,计算:

$$B_i'' = T_i^{-r_i'}, C_i = (g^s)^{r_i'} = g^{r_i}.$$

将计算结果输出至叶子节点 L_i 所在的 Reduce 操作,即 Map 操作输出的键值对为 $(L_i,(B_i'',C_i))$.

- Reduce 操作

以叶子节点 L_i 在 Map 操作输出的键值对 $(L_i,(B_i'',C_i))$ 与该叶子节点到根节点路径上所有非叶子节点 I_j 在 Map 操作输出的键值对$(L_i,s_{i,j})$为 Reduce 操作的输入,令 $s_i' = 1 + \sum_{j=1}^{|parents(L_i)|} s_{i,j}$ (其中,$parents(L_i)$表示根节点到叶子节点 L_i 路径上所有非叶子节点集合),计算 $B_i' = g^{as_i'} B_i'' = g^{as_i'} T_i^{-r_i'}$,该叶子节点 L_i 在 Reduce 操作输出的键值对为 $(L_i,(B_i',C_i))$.

用户客户端将 $\{B_i'\}_{i \in T_{LeafNode}}$ 下载至本地,计算 $B_i = (B_i')^s = g^{as_i} T_i^{-r_i}$ (其中,$s_i = s_i' \cdot s$),将密文子项 $C,C',C_0,\{B_i\}_{i \in T_{LeafNode}}$ 上传至云服务器,并将密文 CT 交由云服务器存储.数据 M 在共享访问树 T 下的完整密文 CT 如下:

$$CT = \langle T, C = R \cdot e(g,g_2)^{\alpha s}, C' = M \oplus r, C_0 = g^s, \forall i \in T_{LeafNode} : B_i = g^{as_i} T_i^{-r_i}, C_i = g^{r_i} \rangle.$$

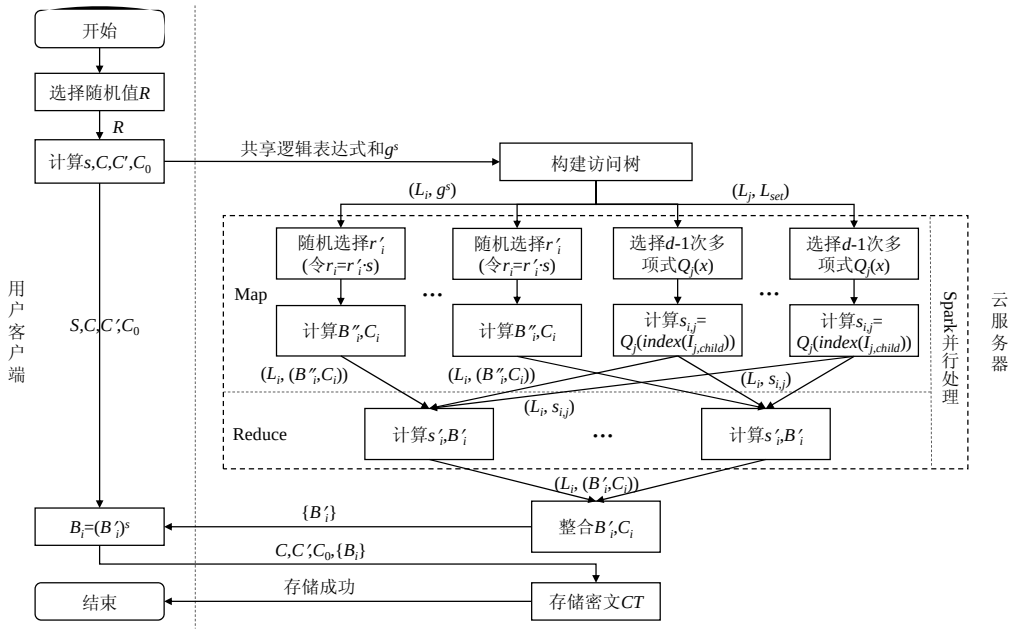

Fig.3 Flow chart of encryption

图 3 加密流程图

(4) 解密:*Decrypt*(*CT*,*TK*,*DK*)

解密时,先将大部分解密工作交由云服务器完成,再由用户客户端完成最后的解密工作.

- 云服务器解密

若用户私钥属性集合 S 不满足 T,则直接输出$\perp$;否则,继续下面的计算.

对于共享访问树 T 的叶子节点 x,$i=att(x)$($att(x)$为叶子节点 x 的属性),其解密算法为

$$DecNode(x)=e(B_x,L)e(C_x,K_i)=e(g^{aQ_x(0)}T_i^{-r_x},g^{\beta t})e(g^{r_x},T_i^{\beta t})=e(g,g)^{a\beta tQ_x(0)}.$$

其中,$Q_x(0)=s_x$.

对于共享访问树 T 的非叶子节点,其解密算法如下:

$$\begin{aligned}DecNode(x)&=\prod_{z\in S_x}DecNode(z)^{\Delta_{i,S'_x}(0)},\ \text{where}\ {}^{i=index(z)}_{S'_x=\{index(z):z\in S_x\}}\\&=\prod_{z\in S_x}(e(g,g)^{a\beta tQ_z(0)})^{\Delta_{i,S'_x}(0)}\\&=\prod_{z\in S_x}(e(g,g)^{a\beta tQ_{parent(z)}(index(z))})^{\Delta_{i,S'_x}(0)}\\&=\prod_{z\in S_x}(e(g,g)^{a\beta tQ_x(i)})^{\Delta_{i,S'_x}(0)}\\&=e(g,g)^{a\beta tQ_x(0)},\end{aligned}$$

其中,$\Delta_{i,S}(x)=\prod_{j\in S,j\neq i}\frac{x-j}{i-j}$.

共享访问树 T 的根节点解密为

$$DecNode(root_T)=e(g,g)^{a\beta tQ_{root_T}(0)}=e(g,g)^{a\beta ts}.$$

最后再计算:

$$F=\frac{e(C_0,K)}{DecNode(root_T)}=\frac{e(g^s,(g^{at}g_2^{\alpha})^{\beta})}{e(g,g)^{a\beta ts}}=e(g,g_2)^{\alpha s\beta}.$$

- 用户客户端解密

数据所有者或共享用户将密文子项 C,C'和云服务器计算的结果 F 下载至本地,解密数据 R 的算法如下:

$$Decrypt(CT,DK)=\frac{C}{F^{\beta^{-1}}}=\frac{R\cdot e(g,g_2)^{\alpha s}}{(e(g,g_2)^{\alpha s\beta})^{\beta^{-1}}}=R.$$

计算 $M=C'\oplus H_2(R),s=H_1(R,M)$:若 $C=R\cdot e(g,g_2)^{\alpha s}$ 且 $F=e(g,g_2)^{\alpha s\beta}$,则输出 M;否则,直接输出$\perp$.

5 安全性与性能分析

5.1 安全性分析

定理 4. 对于任意基于访问树 T 的秘密共享方案,都存在一个线性秘密共享方案(M,ρ)与之等价.

证明:不妨假设访问树 T 为二叉树(任意树都可以转换为二叉树).设 T 有 n 个非叶子节点,依次编号为 1,2,…,n;设 T 有 m 个叶子节点,依次编号为 1,2,…,m.为每个非叶子节点 i 选择函数 $f_i(x)=\omega_i x,\omega_i\in Z_p^*$.

设从根节点到叶子节点 i 依次经过的非叶子节点(下面简称路径)为 $i_1,i_2,...,i_{n_i}$,根据定理 1,第 i 个叶子节点的秘密共享数可表示为

$$\begin{aligned}\lambda_i=s_i&=s+\sum_{j=1}^{n_i}f_{i_j}(index(i_{j+1}))\\&=s+\sum_{j=1}^{n}b_{ij}f_j(index(suc(j))),\ \left(b_{ij}=\begin{cases}1,\ j\in\{i_1,i_2,...,i_{n_i}\}\\0,\ \text{else}\end{cases}\right)\\&=s+\sum_{i=1}^{n}\omega_i b_{ij}index(suc(j)),\ (b_{ij}=1\text{时},suc(j)\text{求}j\text{在路径上的后继节点})\\&=s+\sum_{i=1}^{n}\omega_i b_{ij}c_{ij},\ \left(c_{ij}=index(suc(j))=\begin{cases}1,\ suc(j)\text{为左孩子}\\2,\ suc(j)\text{为右孩子}\end{cases}\right)\\&=(1,b_{i1}c_{i1},b_{i2}c_{i2},...,b_{in}c_{in})(s,\omega_1,\omega_2,...,\omega_n)^T\\&=M_i(s,\omega_1,\omega_2,...,\omega_n)^T.\end{aligned}$$

令 $\rho(M_i)=att(i), M=(M_1,M_2,...,M_m)^T=\begin{bmatrix}1 & ... & b_{1n}c_{1n}\\ ... & ... & ...\\ 1 & ... & b_{mn}c_{mn}\end{bmatrix}$.

故对于任意基于访问树 T 的秘密共享方案,存在一个线性秘密共享方案(M,ρ)与之等价. □

定理 5. 如果 Waters 提出的 CP-ABE 方案[14]达到针对性 CPA 安全,那么所提出的支持快速加密的 CP-ABE 方案在随机预言模型下达到 RCCA 安全[21].

证明:假设攻击者 A 在针对性 RCCA 安全模型中,能在多项式时间内以不可忽略的优势ε攻破本文方案,那么可以构造一个模拟器(或敌手)B,它同样能在针对性 CPA 安全模型中以不可忽略的优势ε攻破 Waters 的方案,而该方案已被证明在 decisional q-parallel BDHE 假设下是安全的. □

Init:模拟器 B 激活敌手 A,A 选择一个挑战访问树 T^*,B 根据定理 4 及其提供的转换方法,将 T^*转换成访问结构(M^*,ρ^*)并传递给 Waters 方案的挑战者,作为其希望被挑战的访问结构.

初始化:模拟器 B 获取 Waters 产生的公钥 $PK=g,e(g,g)^{\alpha},g^{a},\{T_i\}_{i\in U}$,将其作为系统公钥发送给 A.

阶段 1:模拟器 B 初始化空表 T,T_1,T_2,一个空的集合 D,一个整数 $j=0$,攻击者能完成如下查询.

- $H_1(R,M)$:若(R,M,s)已经在 T_1 中,返回 s;否则,选择一个随机值 $s\in Z_p^*$,将(R,M,s)记录在 T_1 中并返回 s;
- $H_2(R)$:若(R,r)已经在 T_2 中,返回 r;否则,选择一个随机值 $r\in\{0,1\}^k$,将(R,r)记录在 T_2 中并返回 r;
- *Create*(S):B 设定 $j:=j+1$,采用下面方式中的进行处理:
  - 如果 S 满足 T^*,则按如下方式选择一个"假"转换密钥:
    选择一个随机数 $d\in Z_p^*$,运行 *KeyGen*$((d,PK),S)$获取 SK',令 $TK=SK',SK=(d,TK)$;
  - 如果 S 不满足 T^*,调用 Waters 的私钥生成算法计算 S 的私钥 $SK'=(PK,K',L',\{K'_x\}_{x\in S})$.算法选择一个随机值 $z\in Z_p^*$,并设置转换密钥 TK 为 $(PK,K=K'^{1/z},L=L'^{1/z},\{K_x\}_{x\in S}=\{K_x'^{1/z}\}_{x\in S})$,私钥为$(z,TK)$;

  最后,将(j,S,SK,TK)存储在 T 中,并将 TK 返回给 A;
- *Corrupt*(i):A 不能询问任何满足访问树 T^*的属性集所相对应的私钥.如果 T 中存在第 i 个元素,B 将获取该元素(i,S,SK,TK)并设 $D:=D\cup\{S\}$,将 SK 返回给 A;若不存在,则返回⊥;
- *Decrypt*(i,CT):作为输入参数的密文 CT 都已经半解密.B 和 A 知道所有私钥的转换密钥,因此两者都可以对所有密文进行半解密.设 $CT=(C_0,C_1,C_2)$为访问树 T^*对应的密文,从 T 中获取记录(i,S,SK,TK),若没有该记录或 S 不满足 T^*,将返回⊥给 A.当第 i 个密钥不满足挑战访问树 T^*时,按如下方式处理.
  1. 解析 $SK=(z,TK)$,计算 $R=C_0/C_2^z$;
  2. 从 T_1 中获取记录(R,M_i,s_i),如果该记录不存在,将⊥返回给 A;
  3. 如在集合 T_1 中存在 $y\neq x$,使得记录(R,M_y,s_y)和(R,M_x,s_x),有 $M_y\neq M_x,s_y=s_x$,则 B 终止仿真;
  4. 否则,获取从 T_2 中获取记录(R,r),若不存在,B 输出⊥;
  5. 对于每个密钥 i,测试 $C_0=R\cdot e(g,g)^{\alpha s_i},C_1=M_i\oplus r,C_2=e(g,g)^{\alpha s_i/z}$ 是否成立;
  6. 通过了上述测试,输出消息 M_i;否则,输出⊥;

  当密钥 i 满足挑战访问树 T^*时,按如下方式处理:
  1. 解析 $SK=(d,TK)$,计算 $\beta=C_2^{1/d}$;
  2. 对于 T_1 中的每个(R_i,M_i,s_i),测试 $\beta=e(g,g)^{s_i}$ 是否成立;
  3. 若都不成立,B 向 A 输出⊥;
  4. 若成立的数量大于 1,则 B 终止仿真;
  5. 否则,设(R,M,s)唯一成立,从 T_2 中获取记录(R,r),若不存在,则 B 输出⊥;
  6. 测试 $C_0=R\cdot e(g,g)^{\alpha s},C_1=M\oplus r,C_2=e(g,g)^{ds}$ 是否成立;
  7. 当所有测试通过时,输出 M;否则,输出⊥.

挑战:A 提交两个等长的消息 $(M_0^*,M_1^*)\in\{0,1\}^{2\times k}$,$B$ 作如下处理.

1. B 选择随机值 $(R_0,R_1)\in G_T^2$,将其传递给 Waters 挑战者获取与访问结构(M^*,ρ^*)相关联的密文:

$$CT=(C,C',\{C_i\}_{i\in[1,l]});$$

2. B 选择一个随机值 $C''\in\{0,1\}^k$;
3. B 向 A 发送挑战密文 $CT^*=(C,C',C'',\{C_i\}_{i\in[1,l]})$.

阶段 2:敌手 A 重复阶段 1 中的查询,若解密查询的响应值是 M_0^* 或 M_1^*,B 用消息 Test 响应.

猜测:A 要么输出 1 或 0,要么退出.无论哪种情况,B 都不予理睬.B 检索表 T_1(或 T_2),如果 $R_b(b\in[0,1])$仅出现一次,则输出 b 作为它的猜测值;否则,随机选择 0 或 1 作为它的猜测值.

敌手赢得游戏的优势被定义为 $\Pr[b'=b]-1/2$.

5.2 性能分析

下面着重讨论本文方案的计算性能和存储性能,并且与 Green 等人所提出的 CP-ABE 外包方案[11]作对比.

(1) 计算性能

在 CP-ABE 算法中,由于计算开销最大的依次是双线性对运算 B、指数运算 E,故采用这 2 项指标来衡量方案的计算性能.

生成用户私钥时,授权中心只需计算密钥子项 K,L 和 T,计算代价为 $4E$;而云集群针对用户的每一个属性执行一次指数运算,计算代价为$|S|E(|S|$表示用户属性数量).

数据加密时,用户客户端需计算 s,r,C,C_0,计算代价为 $4E(H(\cdot)$需执行一次指数运算),针对共享访问树的每个叶子节点,还需执行一次指数运算,故用户客户端的计算代价为$(4+|T_L|)E(|T_L|$表示共享访问树叶子节点数量);而云集群针对共享访问树的每个叶子节点,Map 操作需要执行两次指数运算,Reduce 操作需要执行一次指数运算,故云集群的计算代价为 $3|T_L|E$.

数据解密时,用户客户端的计算代价为 $5E$;在共享访问树叶子节点的逻辑关系都为“AND”的情况下,云集群针对每个叶子节点需计算两次双线性对运算,针对每个非叶子节点需计算两次指数运算,计算 F 时需一次双线性对运算,计算代价为$(1+2|T_L|)B+(2|T_L|-2)E$.

本文方案与 Green 的方案[11]在授权中心和用户客户端的计算开销对比见表 1:本文方案的私钥生成计算开销恒定,仅需 4 次指数运算,而数据加密时的计算开销与 Green 的方案[11]相比减少了 3/4.

Table 1 Comparison of computing time in authorization center and user client

表 1 授权中心和用户客户端计算开销对比

方案	私钥生成	数据加密	数据解密
Green 方案[11]	$(3+2\|S\|)E$	$(4+4\|T_L\|)E$	$5E$
本文方案	$4E$	$(4+\|T_L\|)E$	$5E$

(2) 存储性能

在所提出的方案中,将用户转换密钥和所有的密文数据交由云集群存储,而用户仅需要秘密保存一个最终解密密钥(占用一个群元素空间),减轻了用户对密钥的管理负担.此外,该方案的数据密文长度与 Green 等人所提出的方案[11]相同,故在云集群中所占存储空间与 Green 的方案相同.

6 实验分析

为了评估本文方案在 Spark 并行处理模式下的计算性能,利用双线性对加密库(http://crypto.stanford.edu/pbc/)和 CP-ABE 基础开发包(http://acsc.csl.sri.com/cpabe/),并使用 Java 语言,分别针对本文方案、Proxy 方案(即本文方案在外包计算时不使用 Spark 集群而是使用单台性能较好的服务器)和 Green 的方案[11]编写 CP-ABE 算法并进行性能实验.从素数阶群 $y^2=x^3+x$ 中选取双线性群 G 和 G_T,群 G 和 G_T 的元素长度均为 512 位.该实验所使用的虚拟机 CPU 配置和系统均为:Intel(R) Xeon(R) CPU(E5-2620 2.0GHZ),系统 CentOS6.5 64 位.实验环境包

括 Spark 集群(主节点:4 个 CPU,内存 8G;10 台数据节点:2 个 CPU,内存 8G)、1 台服务器(4 个 CPU,内存 8G)和 1 台用户客户端(1 个 CPU,4G 内存).

私钥生成、加密和解密的性能实验分别针对本文方案、Proxy 方案和 Green 方案各进行 25 轮,每一轮实验所使用的数据、用户属性集合和密文共享访问策略均相同(为了方便不同方案之间的比较,共享访问策略的逻辑关系均使用“AND”),每轮实验均测试 50 次,并以其平均值为最终实验结果,共享访问策略属性数量和用户属性数量依次递增.

生成私钥时,本文方案与 Green 方案的授权中心私钥生成时间和私钥生成总时间分别如图 4、图 6 所示,Proxy 方案和本文方案的服务端转换密钥计算时间如图 5 所示.本文方案授权中心的私钥生成时间恒定,在 178ms 左右,除此之外,其余均随着用户属性数量的增加而线性增长,但本文方案的服务端转换密钥生成时间和私钥生成总时间与对比方案相比明显较低,且随着用户属性数量的增多,两者差异更加明显.

加密时,本文方案与 Green 方案的客户端加密时间和加密总时间分别如图 7、图 9 所示,Proxy 方案和本文方案的服务端加密时间如图 8 所示.无论是客户端和服务端的加密时间,还是加密总时间,都与共享访问策略属性数量呈线性关系,但本文方案在用户客户端和服务端的加密时间都较低,且加密总时间也比 Green 方案低.

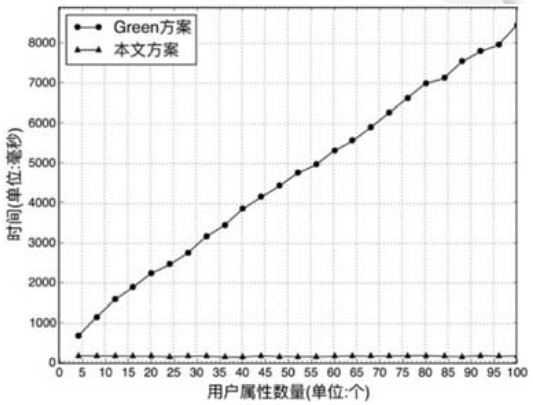

Fig.4 Generation time of private key in authorization center

图 4 授权中心私钥生成时间

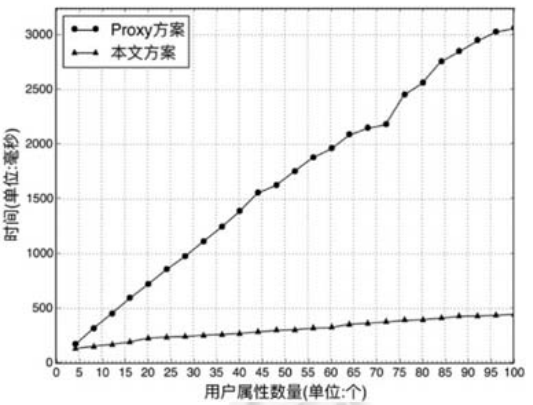

Fig.5 Generation time of transform key in server

图 5 服务端转换密钥生成时间

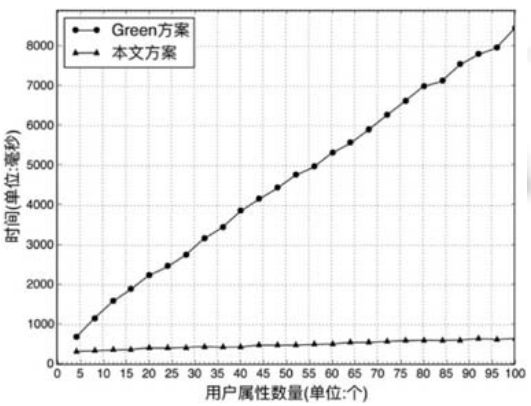

Fig.6 Total generation time of private key

图 6 用户私钥生成总时间

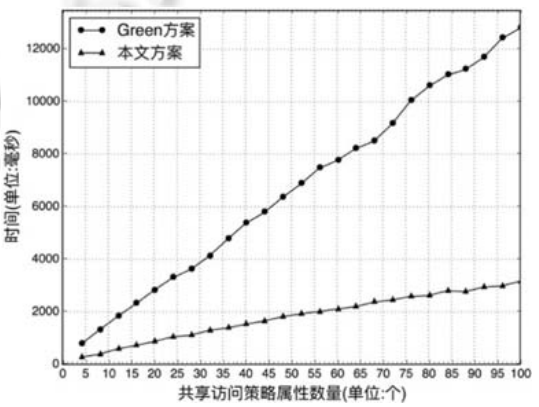

Fig.7 Encryption time in client

图 7 客户端加密时间

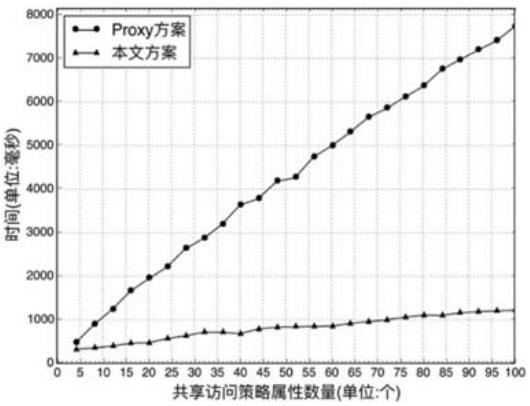

Fig.8 Encryption time in server

图 8 服务端加密时间

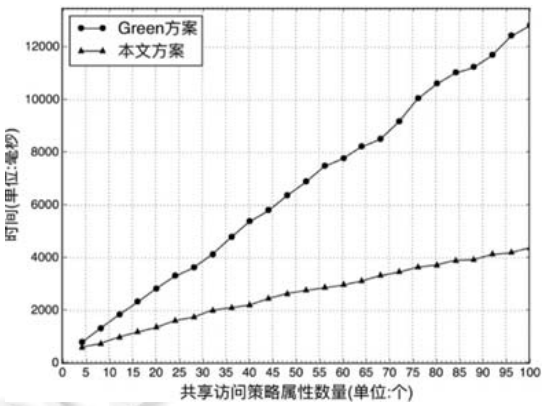

Fig.9 Total encryption time

图 9 加密总时间

解密时,本文方案与 Green 方案的客户端解密时间和服务端解密时间分别如图 10、图 11 所示.两种方案的客户端解密均在 12ms 左右,而服务端解密时间与共享访问策略属性数量呈线性关系,但与 Green 方案相比,本文方案的解密时间较低.

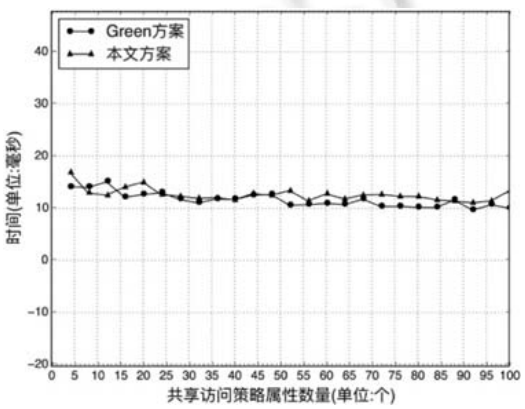

Fig.10 Decryption time in client

图 10 客户端解密时间

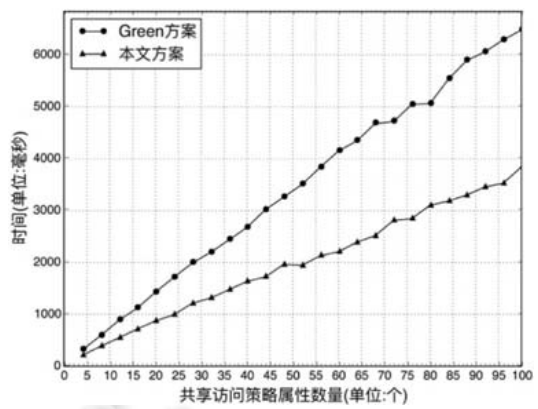

Fig.11 Decryption time in server

图 11 服务端解密时间

7 结束语

计算外包的难点在于实现外包的同时还需要保证数据的安全,尽管现有的一些方案已经将计算外包引入到 CP-ABE 方案的设计之中,但其在安全方面仍然存在缺陷,也没有给出外包加密在云服务器中的并行计算方法,而且还存在授权中心计算量过大、用户保管密钥过多的问题.针对上述问题,提出一种面向公有云的快速加密与共享方案.在该方案中,将用户转换密钥和密文数据外包给云服务器存储,而用户客户端仅需要保存占用一个群元素空间的最终解密密钥,减轻了用户的密钥管理负担.利用 Spark 技术的 Map 操作并行计算用户转换密钥的属性集合,利用 Map 操作和 Reduce 操作并行计算密文共享访问树,其中:Map 操作负责非叶子节点的秘密值分发和叶子节点的密文计算,而 Reduce 操作负责叶子节点的秘密值求和及其幂运算.性能分析表明:该方案不但减少了用户客户端的计算量,而且服务端的计算效率和总计算效率得到显著提升,使其在智能手机、平板这样性能较低的移动设备上也能实现快速加密.安全性分析表明,该方案在随机预言模型下能达到针对性 RCCA 安全.

References:

[1] Ghemawat S, Gobioff H, Leung ST. The Google file system. In: Juels A, Wright RN, Vimercati SDC, eds. Proc. of the 19th ACM Symp. on Operating Systems Principles. Alexandria: ACM, 2003. 29−43. [doi: 10.1145/945445.945450]

[2] Sahai A, Waters B. Fuzzy identity based encryption. In: Proc. of the Advances in Cryptology, Eurocrypt. LNCS: Springer-Verlag, 2005. 457−473. [doi: 10.1007/11426639_27]

[3] Lee CC, Chung PS, Hwang MS. A survey on attribute-based encryption schemes of access control in cloud environments. Int'l Journal of Network Security, 2013,15(4):231−240.

[4] Qiao Z, Liang SW, Davis S, Jiang H. Survey of attribute based encryption. In: Proc. IEEE/ACIS Int'l Conf. on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing. Austria: IEEE, 2014. 1−6. [doi: 10.1109/SNPD.2014.6888687]

[5] Feng CS, Qin ZG, Yuan D. Techniques of secure storage for cloud data. Chinese Journal of Computers, 2015,38(1):150−163 (in Chinese with English abstract).

[6] Feng CS, Qin ZG, Yuan D, Qing Y. Key techniques of access control for cloud computing. Acta Electronica Sinica, 2015,43(2):312−319 (in Chinese with English abstract).

[7] Goyal V, Pandey A, Sahai A, Waters B. Attribute-Based encryption for fine-grained access control of encrypted data. In: Juels A, Wright RN, Vimercati SDC, eds. Proc. of the 13th ACM Conf. on Computer and Communications Security (CCS 2006). Alexandria: ACM, 2006. 89−98.

[8] Bethencourt J, Sahai A, Waters B. Ciphertext-Policy attribute-based encryption. In: Proc. of the 2007 IEEE Symp. on Security and Privacy. Washington: IEEE Computer Society, 2007. 321−334. [doi: 10.1109/SP.2007.11]

[9] Dan B, Franklin M. Identity-Based encryption from the Weil pairing. SIAM Journal on Computing, 2003,32(3):586−615. [doi: 10.1137/S0097539701398521]

[10] Dan B, Lynn B, Shacham H. Short signatures from the Weil pairing. Journal of Cryptology, 2004,17(4):297−319. [doi: 10.1007/s00145-004-0314-9]

[11] Green M, Hohenberger S, Waters B. Outsourcing the decryption of ABE ciphertexts. In: Proc. of the 20th Usenix Conf. on Security. San Francisco: ACM, 2011.

[12] Beimel A. Secure schemes for secret sharing and key distribution [Ph.D. Thesis]. Haifa: Israel Institute of Technology, 1996.

[13] Karchmer M, Wigderson A. On span programs. In: Proc. of the Structure in Complexity Theory Conf. San Diego: Springer-Verlag, 1993. 102−111. [doi: 10.1109/SCT.1993.336536]

[14] Waters B. Ciphertext-Policy attribute-based encryption: An expressive, efficient, and provably secure realization. In: Proc. of the Public Key Cryptography (PKC 2011). Berlin: Springer-Verlag, 2011. 53−70. [doi: 10.1007/978-3-642-19379-8_4]

[15] Zhou Z, Huang D. Efficient and secure data storage operations for mobile cloud computing. In: Proc. of the 8th Int'l Conf. on Network and Service Management. Austria: IEEE, 2012. 37−45.

[16] Li JW, Jia CF, Li J, Chen XF. Outsourcing encryption of attribute-based encryption with MapReduce. In: Proc. of the 14th Int'l Conf. on Information and Communications Security. Berlin: Springer-Verlag, 2012. 191−201. [doi: 10.1007/978-3-642-34129-8_17]

[17] Lai J, Deng RH, Guan C, Weng J. Attribute-Based encryption with verifiable outsourced decryption. IEEE Trans. on Information Forensics and Security, 2013,8(8):1343−1354.

[18] Qin B, Deng R, Liu S, Ma S. Attribute-Based encryption with efficient verifiable outsourced decryption. IEEE Trans. on Information Forensics and Security, 2015,10(7):1384−1393.

[19] Lin S, Zhang R, Ma H, Wang M. Revisiting attribute-based encryption with verifiable outsourced decryption. IEEE Trans. on Information Forensics & Security, 2015,10(10):2119−2130.

[20] Huang Q, Yang Y, Wang L. Secure data access control with ciphertext update and computation outsourcing in fog computing for Internet of things. IEEE Access, 2017,5(99):12941−12950.

[21] Canetti R, Krawczyk H, Nielsen JB. Relaxing chosen-ciphertext security. In: Boneh D, ed. Proc. of the Annual Int'l Cryptology Conf. Berlin: Heidelberg, 2003. 565−582.

附中文参考文献:

[5] 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.

[6] 冯朝胜,秦志光,袁丁,卿昱.云计算环境下访问控制关键技术.电子学报,2015,43(2):312−319.

罗王平(1993－),男,硕士生,主要研究领域为云计算,大数据安全.

冯朝胜(1971－),男,博士,教授,CCF 高级会员,主要研究领域为网络与信息安全,云计算,大数据安全.

邹莉萍(1994－),女,硕士生,主要研究领域为信息安全，云计算,大数据安全.

袁丁(1967－),男,博士,教授,主要研究领域为密码学,信息安全.

吴唐美(1997－),女,本科生,主要研究领域为云计算,大数据安全.

李敏(1978－),女,博士,副教授,主要研究领域为隐私保护,智能学习.

王广杰(1976－),男,副研究员,主要研究领域为地理信息系统.