计算机科学
COMPUTER SCIENCE

基于懒惰模式密文更新的 CP-ABE 属性变动方案

雷雪娇，王银龙，努尔买买提·黑力力

引用本文

雷雪娇，王银龙，努尔买买提·黑力力. 基于懒惰模式密文更新的 CP-ABE 属性变动方案[J]. 计算机科学, 2022, 49(10): 327-334.

LEI Xue-jiao, WANG Yin-long, Nurmamat HELIL. Lazy-mode Ciphertext-update Based Approach for CP-ABE Attribute Change[J]. Computer Science, 2022, 49(10): 327-334.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.211000189

基于懒惰模式密文更新的 CP-ABE 属性变动方案

雷雪娇　王银龙　努尔买买提·黑力力
新疆大学数学与系统科学学院　乌鲁木齐 830017
(leixuejiao866@163.com)

摘　要　密文策略属性基加密(Ciphertext-Policy Attribute-Based Encryption,CP-ABE)可用于实现云计算环境下数据的安全共享。然而,在 CP-ABE 中用户属性的变动(属性撤销和属性添加)是一个棘手的问题。属性变动一般由代理服务器对相关密文进行二次加密和更新用户密钥来实施,而实施属性变动时,需要更新与将发生变动的属性相关的所有密文。文中提出了基于懒惰模式密文更新的用户属性变动方案,该方案通过分析用户(在属性撤销前或属性添加后)对属性变动相关密文是否具有访问能力,来判断是否需要更新密文,尽可能缩小需要更新的密文范围以及减少密文更新的次数,在保留原有 CP-ABE 方案安全特性的情况下,通过避免不必要的密文更新以及缩短密文长度的方式来提高方案的有效性。最后,通过小型实验验证了所提方案的正确性。

关键词:密文策略属性基加密;属性撤销;属性增加;密文更新;懒惰模式
中图法分类号　TP309

Lazy-mode Ciphertext-update Based Approach for CP-ABE Attribute Change

LEI Xue-jiao,WANG Yin-long and Nurmamat HELIL
College of Mathematics and System Science,Xinjiang University,Urumqi 830017,China

Abstract　Ciphertext-policy attribute-based encryption(CP-ABE) can be used to realize secure data sharing in cloud computing environments. However,user attribute change(attribute revocation and addition) in CP-ABE is a tricky problem. Generally,attribute change is realized via the proxy server's secondary encryption of ciphertext and key update. However,when enforcing an attribute change,all ciphertexts related to this attribute should be updated. This paper proposes a user attribute change approach based on lazy-mode ciphertext-update. It analyzes the user's access ability(before attribute revocation or after attribute addition) to the ciphertexts involved in attribute change and determines if these ciphertexts need to be updated,minimizing the scope of the ciphertexts that need to be updated and reducing the number of updates. This approach improves its efficiency by avoiding unnecessary ciphertext updates and shortening the ciphertext while preserving the original security features of the CP-ABE. Finally,a small-size test is conducted to verify the correctness of the proposed approach.

Keywords　Ciphertext-policy attribute-based encryption,Attribute revocation,Attribute addition,Ciphertext update,Lazy-mode

1　引言

Sahai 等提出的 CP-ABE 方案[1-3]中,数据拥有者可以自由地定义访问结构,访问控制策略在表达方面灵活性高。然而,在 CP-ABE 中,用户的属性随时可能发生变动,即属性撤销与属性增加。属性变动是 CP-ABE 系统中的重要问题及挑战。撤销用户的部分属性或给用户增加一些新属性的目的是使用户失去或获取这些属性相关部分数据"读"的访问权限。文献[4-5]提出了有效的 CP-ABE 属性撤销方案,研究了集中撤销恶意用户的解密能力,但不影响其他诚实用户的解密能力,并且提高了撤销的效率;文献[6]利用双重加密机制实现细粒度的属性撤销,但无法抵抗合谋攻击;文献[7-9]也讨论了关于属性撤销的问题。Liu 等首次提出通过树型访问结构来达到属性撤销的目的[7],方案将属性撤销与细粒度访问控制相结合,利用代理服务器实现重加密,减小终端用户的负担。在文献[8]中,原始数据被分成若干块,然后被发布到云中存储。当发生属性撤销时,数据所有者只需要检索与撤销与属性相关的一个数据块,并对其进行重新加密和重新发布。因此,通过只影响一个数据块而不是整个数据,可以加快属性撤销的实施。文献[9]提出了在直接撤销模式下,可以对用户任意数量的属性进行撤销,解决了已有方案中属性撤销时粒度过粗的问题。

CP-ABE 中,有效实施属性撤销也是一个值得探讨的问题。在 Pirretti 等提出的属性撤销方案[10]中,系统给拟撤销属性相关的密钥提供了一个版本号,即属性撤销时更新密钥版本,撤销属性的用户无法得到更新后的密钥,因此用户失去

到稿日期:2021-10-25　返修日期:2022-04-06
基金项目:国家自然科学基金(61862059,61562085)
This work was supported by the National Natural Science Foundation of China(61862059,61562085).
通信作者:努尔买买提·黑力力(nur924@sina.com)

相应的属性后，利用这种方法不能实现即时的属性撤销。文献[11]通过引入属性组密钥，对密钥进行动态修改，实现了属性立即撤销的要求。Li 等提出了具有属性撤销的可搜索 CP-ABE 方案[12]，该方案支持用户属性的撤销，并在属性撤销过程中将密文更新的大部分工作转移给云服务提供商来完成。文献[13]通过设计高效的重加密算法，引入属性撤销列表，实现了细粒度的属性直接撤销。

在属性变动过程中，保证抗合谋攻击也是一个挑战。Li 等提出了支持抗合谋攻击的属性撤销方案[14-15]，该方案中用户的私钥分为两部分，即私有密钥和密钥加密密钥，当用户的属性被撤销时，管理器会更新其他用户的密钥加密密钥，阻止撤销用户与未撤销用户合谋攻击。

本文在现有 CP-ABE 方案[14]的基础上，在不降低方案的安全特性的前提下，对属性变动的实施进行优化。属性变动过程采用密文更新懒惰模式，利用属性密文组和用户密文组的概念，尽可能缩小需要更新的密文范围以及减少密文更新的次数，避免不必要的密文更新，提高属性变动的实施效率。安全证明表明，所提方案在选择明文攻击下是安全的。

2 预备知识

定义 1(双线性对) p 是素数，G_1，G_T 是阶为 p 的乘法循环群，$G_1 \times G_1 \rightarrow G_T$ 是双线性对，它满足下述性质：

(1) 双线性：$\forall g_1, g_2 \in G_1, a, b \in Z_p$，$e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$。

(2)非退化性：$e(g_1, g_2) \neq 1$。

(3)可计算性：$e(g_1, g_2)$是有效的计算。

定义 2(CDH 假设) 给定三元组$(g, g^{x_1}, g^{x_2}) \in G^3$，$x_1$，$x_2 \in Z_p^*$ 并且是未知的，需要求出 $g^{x_1 x_2}$ 的值，若挑战者输出 $g^{x_1 x_2}$ 的概率 $Adv^{\mathrm{CDH}} = |Pr[A(^{x_1}, g^{x_2}) = g^{x_1 x_2}]| \leqslant \varepsilon$，其中 ε 是可忽略的，则 CDH 问题是困难的。

定义 3(属性密文组) 设 $A = \{\lambda_1, \lambda_2, \cdots, \lambda_m\}$是系统中所有属性的集合，$C = \{C_1, C_2, \cdots, C_n\}$ 是全体密文集合，称在访问树结构 τ 中有属性λ_i 的所有密文集合 CG_{λ_i} ($CG_{\lambda_i} \in 2^C$)为属性 λ_i 的属性密文组。

定义 4(用户密文组) 设 $\omega_u = \{\lambda_1, \lambda_2, \cdots, \lambda_p\}$是用户 u 的属性集合，$\bar{C} = \{C_1, C_2, \cdots, C_l\}$是 ω_u 中所有属性的属性密文组的并集。称用户 u 在 $\bar{C}$ 中能访问的密文集 CG_u ($CG_u \in 2^{\bar{C}}$) 为用户密文组。

表 1 列出了文中使用的符号及其含义。

表 1 符号表

Table 1 Notations

Λ	撤销的属性集合
Ψ	增加的属性集合
Γ	用户的集合
A	系统属性集合
Φ	A 中属性的属性用户组的集合
PK(PublicKey)	公共密钥
MK(MasterKey)	主密钥
MPK(Manager PublicKey)	管理器公钥
MMK(Manager MasterKey)	管理器主密钥
KEK(Key EncryptingKey)	密钥加密密钥
DSK(Decryption SecretKey)	解密密钥

3 算法流程及安全模型

3.1 算法流程

(1)系统初始化：输入安全参数 κ，输出 MK 和 PK。

(2)管理器初始化：输入 PK，输出 MMK 和 MPK。

(3)密钥生成算法：输入 MK，PK，MMK，MPK，Γ 和 ω_{u_a}，输出用户的 DSK 和 kek。

(4)密钥加密密钥生成算法：输入属性集 ω_{u_a}，输出 KEK 并发送给用户。

(5)加密算法：输入 PK，MPK，信息 M，访问结构 τ，输出密文 CT。

(6)重加密算法：将密文 CT 和属性用户组 Φ 作为输入，将重加密得到的密文$\overline{CT}$和头部信息 Hdr 作为输出。

(7)属性用户组解密算法：输入头部信息 Hdr、用户二叉树，只要用户未撤销此属性，则能输出属性用户组密钥 k_i，否则输出错误符号$\perp$。

(8)信息解密算法：输入$\overline{CT}$，DSK，PK，MPK，Hdr，kek，KEK，输出 M。

3.2 安全模型

定义 5 在安全模型中，表现出的攻击是现有用户与撤销用户合作产生的合谋攻击，因此挑战者需要对两种类型的私钥进行询问。第一种类型是：用户 u_1 的属性集合 ω_{u_1} 满足挑战的访问树结构 τ^*，但 ω_{u_1} 中的挑战属性 λ^* 已经被撤销；第二种类型是：用户 u_2 的属性集合 ω_{u_2} 不满足需要挑战的访问树 τ^*，但 ω_{u_2} 中的挑战属性 λ^* 未被撤销。

(1)初始化：挑战者选择了挑战的访问策略 τ^* 及挑战的属性 λ^*，并将其发送给模拟器，其中 λ^* 是满足 τ^* 的必要属性值之一。

(2)系统建立：模拟器调用系统初始化，计算出系统的公共参数 PP 和主密钥 MK；然后模拟器再执行属性管理器(Attribute Manager，AM)初始化，计算出 AM 的公钥 MPK 和主密钥 MMK，同时模拟器更新挑战属性 λ^* 的密钥对，得到更新的管理器公钥$\overline{MPK}$和主密钥$\overline{MMK}$，最后模拟器将 PP，MPK 和$\overline{MPK}$发送给挑战者，保留 MK，MMK 和$\overline{MMK}$。

(3)阶段 1：挑战者整体上对两种类型的密钥进行询问。两种类型的询问分别为 $Type_1$ 和 $Type_2$。

1)$Type_1$ 类型的私钥询问：用户 u_1 的属性集合 ω_{u_1} 满足挑战的访问树结构 τ^*，但 ω_{u_1} 中的挑战属性 λ^* 已经被撤销，模拟器运行 KeyGen 算法，获得 DSK_1 和 KEK_1 发送给挑战者。

2)$Type_2$ 类型的私钥询问：用户 u_2 的属性集合 ω_{u_2} 不满足需要挑战的访问树 τ^*，但 ω_{u_2} 中的需要挑战属性 λ^* 未被撤销；模拟器运行 KeyGen 算法，获得 DSK_2 和 KEK_2 并将其发送给挑战者。

(4)挑战：若阶段 1 结束，输出两个等长的明文 M_0 和 M_1，模拟器随机选择一个值 $\mu \in \{0, 1\}$。运行 Encrypt 算法，计算挑战密文$\langle Hdr^*, CT_\mu^* \rangle$，并将其发送给挑战者。

(5)阶段 2：类似于阶段 1。

(6)猜测：挑战者输出一个猜测值 $\bar{\mu} \in \{0, 1\}$，若 $\bar{\mu} = \mu$，则挑战者赢得此游戏。挑战者在此游戏中获胜的优势为 $Adv = \left| Pr[\bar{\mu} = \mu] - \frac{1}{2} \right|$。

定义 6 若在任意概率多项式时间内，挑战者能够利用不可忽略的优势赢得此游戏，则称本文方案是安全的。

4 属性变动方案的工作流程

属性变动方案的工作流程如图 1 所示。

(1)系统初始化：由可信权威机构(Trusted Authority, TA)执行，过程如下：

1)假定 G_1 和 G_T 是素数阶 p 的乘法循环群，g 是 G_1 的生成元，$e:G_1\times G_1\rightarrow G_T$ 是双线性映射。

2)随机选择 $\alpha\in Z_p^*$，获得主密钥 $MK=\alpha$，计算 $e(g,g)^\alpha$，公钥 $PK=\{g,e(g,g)^\alpha\}$。

(2)管理器初始化：对每一个 $\lambda_i\in A(1\leqslant i\leqslant m)$，随机生成 $t_1,t_2,\cdots,t_m\in Z_p^*$，计算 $T_i=g^{\frac{1}{t_i}}(1\leqslant i\leqslant m)$。管理器的主密钥 $MMK=\{t_1,t_2,\cdots,t_m\}$，管理器的公钥 $MPK=\{T_1,T_2,\cdots,T_m\}$。

(3)密钥生成：由 TA 执行此算法，并生成每一个用户对应的私有密钥。

1)随机选择 $r\in Z_p$，计算 $D_0=g^{\alpha r}$，对每一个 $\lambda_i\in\omega_{u_a}$ ($i=1,2,\cdots,m;a=1,2,\cdots,n$)，计算 $D_i=g^{rt_i}$，发送相应的私钥给每个用户。

$DSK=\{D_0=g^{\alpha-r},\forall\lambda_i\in\omega_{u_a},D_i=g^{rt_i}\}$，其中 r 是用户个性化参数。

2)对每一个 $\lambda_i\in A$，随机选择 $k_i\in Z_p^*$，TA 计算 $kek_i=g^{\alpha t_i k_i^{-1}}$，并将$\{\lambda_i,kek_i\}$添加到 KEK。执行完上述操作后，TA 将每个属性 $\lambda_i\in A$ 对应的属性用户组 Φ_i 发送给 AM。

(4)KEK 生成算法：AM 收到属性用户组后，为 Γ 中用户生成一个二叉 KEK 树，用于分发用户的属性用户组密钥，在 KEK 树中，每一个节点持有一个唯一值 v_q，将每个叶子节点分配给用户。对每个用户 $u_a\in\Gamma$，计算从叶子节点到根节点的路径节点，用 $path(u_a)$ 表示。

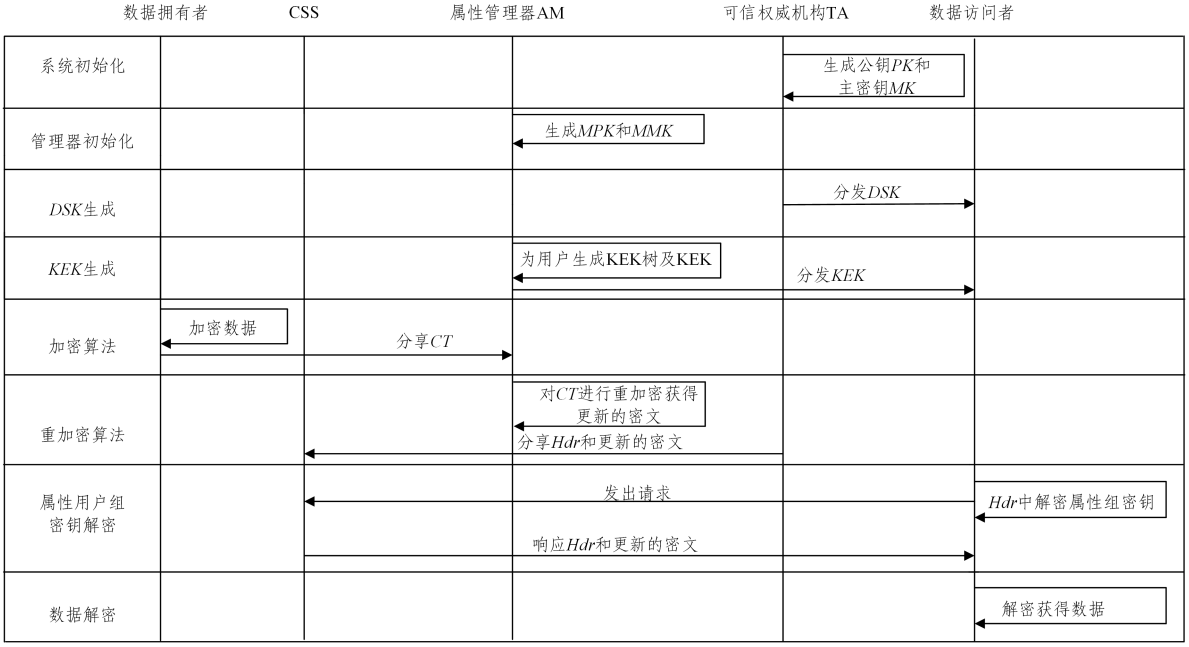

图 1 系统框架

Fig. 1 System framework

1)对于每一个属性 $\lambda_i\in A$，AM 在 KEK 树中定义了相应的最小覆盖集，并且覆盖了与在 Φ_i 中用户相关的所有叶子节点，用函数 $node(\Phi_i)$ 表示。

2)对每个属性 $\lambda_i\in\omega_{u_a}$，执行交操作 $\varphi=node(\Phi_i)\cap path(u_a)$，计算 $KEK_i=kek_i^{\frac{1}{v_q}}=g^{\frac{\alpha t_i k_i^{-1}}{v_q}}$，用$\{\lambda_i,v_q,kek_i,KEK_i\}$取代$\{\lambda_i,kek_i\}$。

(5)加密算法：数据拥有者(DO)制定一个访问树 τ，步骤如下：

1)第一层加密：选择随机数 $h\in Z_p^*$，计算 $C_0=Me(g,g)^{\alpha h}$。

2)第二层加密：设定 τ 的根节点的值为 h，标记所有的子节点为未分配，标记根节点为已分配；对每个未分配的非叶子节点执行以下操作。

若符号为 $\wedge$，并且它的子节点被标记为未分配，则利用模加机制给子节点分配一个值，为此，为除最后一个子节点外的每个子节点赋予一个随机值 $h_j(1\leqslant h_j\leqslant p-1)$，最后一个子节点的赋值为 $h_\zeta=h-\sum_{j=1}^{\zeta-1}h_j \bmod p$，并标记这个节点为已赋值。

若符号为 $\vee$，则将每个子节点设置为 h，并标记该节点为已赋值。

3)对每个叶子属性 $\lambda_{i,j}\in Y$，(Y 表示 τ 的叶子属性的集合，$Y_\wedge$ 表示 $\wedge$ 节点的 $\zeta-1$ 个子叶子属性的集合，$Y_\vee$ 表示 $\vee$ 节点的子叶子属性，j 表示 τ 中该叶子属性的索引值)，计算 $C_{i,j}=T_i^{h_j}$。

4)返回密文 $CT=\{\tau,C_0,\forall\lambda_{i,j}\in Y,C_{i,j}\}$。

访问树的秘密分配的具体过程见文献[16]。

(6)重加密算法：AM 收到密文后，根据属性用户组对密文进行重加密，构造过程如下：

1)对于任意的 $\Phi_i \in \Phi$, $\forall \lambda_{i,j} \in Y$,计算 $\bar{C}_{i,j} = C_{i,j}^{k_i}$,则 $\overline{CT} = \{\tau, C_0, \forall \lambda_{i,j} \in Y, \bar{C}_{i,j}\}$。

2)对于密文访问树中的每个属性 λ_i,AM 生成 $node(\Phi_i)$。

3)AM 生成头部信息 $Hdr = \{\forall \lambda_{i,j} \in Y, \langle v_q, E(k_i) = g^{\frac{k_i v_q}{t_i}} \rangle_{v_q \in node(\Phi_i)}\}$。

(7)属性用户组密钥解密:只要用户未被撤销出属性用户组,就能够从 Hdr 中得到属性用户组密钥,即便是用户并没有按时更新自己的密钥。用户首先解密属性集 ω_{u_a} 中所有属性对应的属性用户组密钥,若 $u_a \in \Phi_i$,则解密属性用户组密钥 $k_i = D(Hdr)$,其中 $v_q \in node(\Phi_i)$。

(8)信息解密算法:用户输入自己的属性列表 $\omega_{u_a} = \{\lambda_1, \lambda_2, \cdots, \lambda_\rho\}$,执行解密算法,计算 $M = \frac{C_0 \cdot e(D_0, g) e(D_i, T_i)}{(\prod_{\lambda_i \in \omega_{u_a}} e(kek_i, \bar{C}_{i,j})) \cdot e(KEK_i, E(k_i))}$。

1)对每一个属性 $\lambda_i \in \omega_{u_a}$, $\prod_{\lambda_i \in \omega_{u_a}} e(kek_i, \bar{C}_{i,j}) = \prod_{\lambda_i \in \omega_{u_a}} e(g^{\alpha t_i k_i^{-1}}, T_i^{h_j k_i}) = e(g,g)^{\alpha h}$; $e(KEK_i, E(k_i)) = e(g,g)^{\alpha}$。

2)计算 $e(D_i, T_i) = e(g^{r t_i}, g^{\frac{1}{t_i}}) = e(g,g)^r$。

3)计算 $\frac{C_0}{e(g,g)^{\alpha h}} = M$。

5 属性变动

在文献[8]和文献[14]中,我们发现撤销或增加某个用户的给定属性时,AM 首先会更新相应的属性用户组,然后更新与这个属性相关的密文(指属性密文组中所有的密文)及属性用户组中与此属性相关的其他用户密钥。AM 更新属性密文组中的所有密文时,可能会造成多余的或重复的密文更新。因此,本文根据属性变动的情况以及此属性的属性密文组与用户密文组,得到新的非更新不可的密文集合,仅对此密文集合中的密文进行更新,剩余的密文不必更新,从而减少更新的密文数量并避免同一密文多次不必要的更新。

为了叙述方便,先引进属性变动原子操作的概念。

定义 7(属性变动原子操作) 系统实施一次属性撤销或属性增加时需要进行的所有的密文更新、密钥更新的操作总和被称为属性变动的原子操作。

按照原子操作的本质,属性变动原子操作包含的所有操作要么全部执行,也就意味着属性变动成功实施;要么全部不执行,意味着属性变动实施失败。

下文分不同的情形讨论如何实施属性变动原子操作,以及如何避免不必要的密文更新。

5.1 属性变动的一次原子操作

5.1.1 一个用户撤销一个属性或一个用户增加一个属性

假设一个用户 u 的属性集合 ω_u 中的一个属性需要进行变动(撤销/增加),若 λ 的属性密文组 CG_λ 与撤销(增加)属性 λ 之前(之后)u 的用户密文组 CG_u 有交集,即 $\overline{CG} = CG_\lambda \cap CG_u = \{\overline{c_1}, \overline{c_2}, \cdots, \overline{c_q}\}$,则当 λ 被撤销(增加)时,AM 只需要更新 $\overline{CG}$ 中的密文,没有必要更新 CG_λ 中的全部密文(见图 2)。

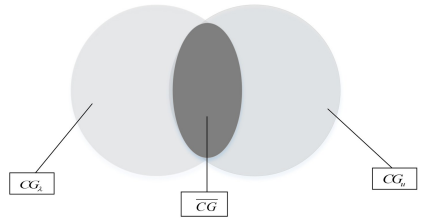

图 2 拟更新密文集(5.1.1)

Fig. 2 Ciphertext sets to be updated(5.1.1)

下文简单假设一个 CP-ABE 系统中的密文集为 $\{c_1, c_2, c_3, c_4\}$,对应的访问树分别为 $\tau_1, \tau_2, \tau_3, \tau_4$(见图 3)。

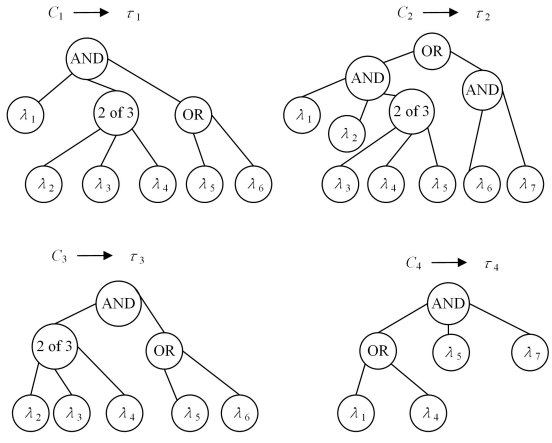

图 3 访问树示例

Fig. 3 Example of access trees

例 1 用户 u 的属性集合 $\omega_u = \{\lambda_1, \lambda_2, \lambda_3, \lambda_6\}$。现对 u 的属性集进行如下变动,以下两项操作是独立进行的。

(1)撤销属性 λ_1,如图 3 所示,$CG_{\lambda_1} = \{c_1, c_2, c_4\}$,而撤销属性 λ_1 之前,用户 u 的用户密文组为 $CG_u = \{c_1, c_3\}$,因此 $CG_{\lambda_1} \cap CG_u = \{c_1\}$,故撤销 λ_1 时,密文 c_1 需要更新。

(2)增加属性 λ_4,如图 3 所示,$CG_{\lambda_4} = \{c_1, c_2, c_3, c_4\}$,增加 λ_4 之后,$CG_u = \{c_1, c_2, c_3\}$,因此 $CG_{\lambda_4} \cap CG_u = \{c_1, c_2, c_3\}$,故增加 λ_4 时,密文 c_1, c_2, c_3 需要更新。

5.1.2 一个用户撤销多个属性或一个用户增加多个属性

假设一个用户 u 的属性集合 ω_u 中的 δ 个属性 $\{\lambda_1, \lambda_2, \cdots, \lambda_\delta\}$ 需要进行变动(撤销/增加),且属性密文组的并集 $\bigcup_{i=1}^{\delta} CG_{\lambda_i}$ 在撤销(增加)δ 属性之前(之后)与 CG_u 有交集,即 $\overline{CG} = (\bigcup_{i=1}^{\delta} CG_{\lambda_i}) \cap CG_u$,则 $\{\lambda_1, \lambda_2, \cdots, \lambda_\delta\}$ 撤销(增加)时,AM 只需要更新 $\overline{CG}$ 中的密文而且各密文仅更新一次,没必要更新 $\bigcup_{i=1}^{\delta} CG_{\lambda_i}$ 中的全部密文(见图 4)。

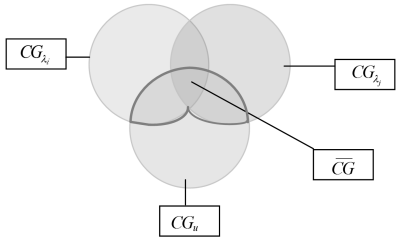

图 4 拟更新密文集(5.1.2)

Fig. 4 Ciphertext sets to be updated(5.1.2)

例 2　用户 u 的属性集合 $\omega_u=\{\lambda_1,\lambda_2,\lambda_3,\lambda_6\}$，现对 u 的属性集进行如下变动。

(1)撤销属性 λ_1 和 λ_2，如图 3 所示，$CG_{\lambda_1}=\{c_1,c_2,c_4\}$，$CG_{\lambda_2}=\{c_1,c_2,c_3\}$，而撤销属性 λ_1 和 λ_2 之前，$CG_u=\{c_1,c_3\}$，因此，$CG_{\lambda_1}\cup CG_{\lambda_2}=\{c_1,c_2,c_3,c_4\}$，$(CG_{\lambda_1}\cup CG_{\lambda_2})\cap CG_u=\{c_1,c_3\}$，故撤销 λ_1,λ_2 时，密文 c_1,c_3 需要更新而且仅需要更新一次。文献[14]中，对于同样的属性撤销需求，需要更新 c_1,c_2,c_3,c_4，而且 c_1,c_2 各更新两次。

(2)增加属性 λ_4 和 λ_7，如图 3 所示，$CG_{\lambda_4}=\{c_1,c_2,c_3,c_4\}$，$CG_{\lambda_7}=\{c_2,c_4\}$，且增加 λ_4 和 λ_7 之后，$CG_u=\{c_1,c_2,c_3\}$，因此，$CG_{\lambda_4}\cup CG_{\lambda_7}=\{c_1,c_2,c_3,c_4\}$，$(CG_{\lambda_4}\cup CG_{\lambda_7})\cap$，即$\overline{CG}=\{c_1,c_2,c_3\}$，故增加 λ_4 和 λ_7 时，密文 c_1,c_2,c_3 需要更新而且仅需要更新一次。文献[14]中，对于同样的属性增加需求，需要更新 c_1,c_2,c_3,c_4，而且 c_2,c_4 各更新两次。

5.1.3　3 多个用户撤销一个属性或多个用户增加一个属性

假设 σ 个用户 $\{u_1,u_2,\cdots,u_\sigma\}$ 的属性集合 $\bigcup_{i=1}^{\sigma}\omega_{u_i}$ 中同一个属性 λ 需要进行变动(撤销/增加)，若 λ 的属性密文组为 CG_λ 与撤销(增加)λ 之前(之后)的各个用户的用户密文组的并集有交集，即$\overline{CG}=(\bigcup_{i=1}^{\sigma}CG_{u_i})\cap CG_\lambda$，则 λ 被撤销(增加)时，AM 只需更新$\overline{CG}$中的密文，没有必要更新 CG_λ 中的全部密文(见图 5)。

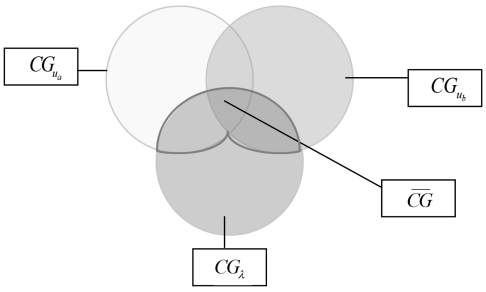

图 5　拟更新密文集(5.1.3)

Fig.5　Ciphertext sets to be updated(5.1.3)

例 3　用户 u_a 的属性集合 $\omega_{u_a}=\{\lambda_1,\lambda_2,\lambda_3,\lambda_6\}$，用户 u_b 的属性集合 $\omega_{u_b}=\{\lambda_1,\lambda_2,\lambda_4,\lambda_6\}$，现对 u_a 和 u_b 的属性集进行如下变动。

(1)撤销用户 u_a 和 u_b 的同一个属性 λ_1，如图 3 所示，$CG_{\lambda_7}=\{c_1,c_2,c_4\}$，对于 u_a,u_b 而言，撤销属性 λ_1 之前，$CG_{u_a}=\{c_1\}$，$CG_{u_b}=\{c_1,c_3\}$，故 $CG_{u_a}\cup CG_{u_b}=\{c_1,c_3\}$，$CG_{\lambda_1}\cap(CG_{u_a}\cup CG_{u_b})=\{c_1\}$，AM 只需要更新密文 c_1。

(2)u_a 和 u_b 增加同一个属性 λ_5，如图 3 所示，$CG_{\lambda_5}=\{c_1,c_2,c_3,c_4\}$，对于 u_a,u_b 而言，增加属性 λ_5 之后，$CG_{u_a}=\{c_1,c_2,c_3\}$，$CG_{u_b}=\{c_1,c_2,c_3\}$，故 $CG_{u_a}\cup CG_{u_b}=\{c_1,c_2,c_3\}$，$CG_{\lambda_5}\cap(CG_{u_a}\cup CG_{u_b})=\{c_1,c_2,c_3\}$，则 c_1,c_2,c_3 需要更新。

5.1.4　多个用户撤销多个属性或多个用户增加多个属性

假设 σ 个用户 $\{u_1,u_2,\cdots,u_\sigma\}$ 的每一个用户 u_j 的属性集合 ω_{u_j} 中有 δ 个属性 $\{\lambda_1,\lambda_2,\cdots,\lambda_\delta\}$ 需要进行变动(撤销/增加)，若 δ 个属性的属性密文组 $\bigcup_{i=1}^{\delta}CG_{\lambda_i}$ 与撤销(增加)δ 个属性之前(之后)的用户密文组 CG_{u_j} 作交集，即$\overline{CG_j}=(\bigcup_{i=1}^{\delta}CG_{\lambda_i})\cap CG_{u_j}$，之后 σ 个用户分别对应$\overline{CG_j}$，取并集为$\bigcup_{j=1}^{\sigma}\overline{CG_j}$。因此，当 σ 个用户撤销(增加)属性时，AM 只需更新$\bigcup_{j=1}^{\sigma}\overline{CG_j}$中的密文，没必要更新$\bigcup_{i=1}^{\delta}CG_{\lambda_i}$ 中的全部密文(见图 6)。以用户为单位，其中 $\overline{CG_j}$ 表示用户 u_j 发生属性变动时最终需要更新的密文。

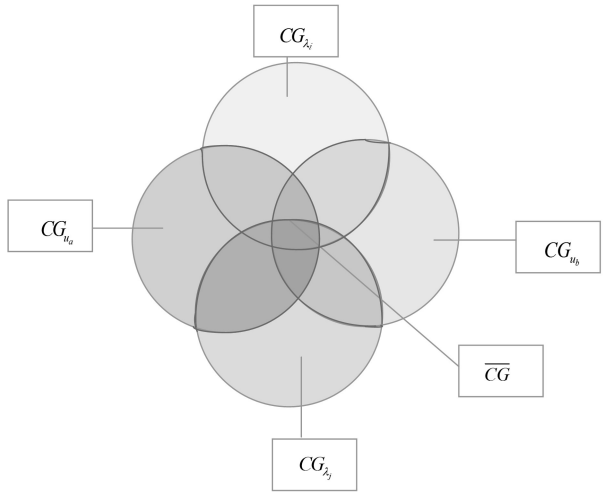

图 6　拟更新密文集(5.1.4)

Fig.6　Ciphertext sets to be updated(5.1.4)

例 4　用户 u_a 的属性集合 $\omega_{u_a}=\{\lambda_1,\lambda_2,\lambda_3,\lambda_6\}$，$u_b$ 的属性集合 $\omega_{u_b}=\{\lambda_1,\lambda_2,\lambda_4,\lambda_6\}$，现对 u_a 和 u_b 属性集进行如下变动。

(1)撤销用户 u_a 和 u_b 属性的 λ_1,λ_6，如图 3 所示，$CG_{\lambda_1}=\{c_1,c_2,c_4\}$和 $CG_{\lambda_6}=\{c_1,c_2,c_3\}$，而 u_a 和 u_b 在撤销属性 λ_1 和 λ_6 之前，各自的用户密文组为 $CG_{u_a}=\{c_1,c_3\}$，$CG_{u_b}=\{c_1,c_3\}$，故 $CG_{\lambda_1}\cup CG_{\lambda_6}=\{c_1,c_2,c_3,c_4\}$，之后$\overline{CG_a}=(CG_{\lambda_1}\cup CG_{\lambda_6})\cap CG_{u_a}=\{c_1,c_3\}$及$\overline{CG_b}=(CG_{\lambda_1}\cup CG_{\lambda_6})\cap CG_{u_b}=\{c_1,c_3\}$，最后$\overline{CG}=\overline{CG_a}\cup\overline{CG_b}=\{c_1,c_3\}$，因此，撤销 λ_1 和 λ_6 时，密文 c_1,c_3 需要更新而且仅需要更新一次。文献[14]中，对于同样的属性撤销需求，需要更新 c_1,c_2,c_3,c_4，而且 c_1,c_2 各更新两次。

(2)u_a 增加属性 λ_5，u_b 增加属性 λ_7，如图 3 所示，$CG_{\lambda_5}=\{c_1,c_2,c_3,c_4\}$和 $CG_{\lambda_7}=\{c_2,c_4\}$，而 u_a 增加属性 λ_5 和 u_b 增加属性 λ_7 之后，各自的用户密文组为 $CG_{u_a}=\{c_1,c_2,c_3\}$，$CG_{u_b}=\{c_1,c_2,c_3\}$。故$\overline{CG_a}=CG_{\lambda_5}\cap CG_{u_a}=\{c_1,c_2,c_3\}$及$\overline{CG_b}=CG_{\lambda_7}\cap CG_{u_b}=\{c_2\}$，$\overline{CG}=\overline{CG_a}\cup\overline{CG_b}=\{c_1,c_2,c_3\}$，因此密文 c_1,c_2,c_3 需要更新。

下面举例补充说明混合的情况：一个属性变动原子操作同时含有撤销和增加。

用户 u_a 撤销属性λ_1 和用户 u_b 增加属性λ_7，如图 3 所示，$CG_{\lambda_1}=\{c_1,c_2,c_4\}$和 $CG_{\lambda_7}=\{c_2,c_4\}$，$u_a$ 撤销属性 λ_1 之前用户密文组为 $CG_{u_a}=\{c_1,c_3\}$，故$\overline{CG_a}=CG_{\lambda_1}\cap CG_{u_a}=\{c_1\}$。$u_b$ 在增加属性 λ_7 之后的用户密文组 $CG_{u_b}=\{c_1,c_2,c_3\}$，取其交集$\overline{CG_b}=CG_{\lambda_7}\cap CG_{u_b}=\{c_2\}$。因此，当发生属性变动时，AM 只需要更新$\overline{CG}=\overline{CG_a}\cup\overline{CG_b}=\{c_1,c_2\}$。

5.2　属性撤销

(1)密钥更新：TA 接收到属性用户组中某个用户离开时

(即某个用户撤销此属性)进行密钥更新的过程,并将更新后属性用户组发送给 AM,AM 进行属性用户组密钥更新的操作如下:

对每个属性 $\lambda_i \in \Lambda(1 \leqslant i \leqslant m)$,AM 更新属性用户组为 Φ_t,同时节点的输出函数 $node()$ 将变成 $node(\Phi_t)$,对任意一个用户 $u_t \in \Phi_t$,AM 执行交操作 $\varphi' = node(\Phi_t) \cap path(u_t)$,且随机选择 $\chi_i \in Z_p$ 和 $k_t' \in Z_p^*$ $(k_t' \neq k_t)$,使得 $t_i' = t_i \cdot \chi_i$,$kek'_i = g^{\alpha t_i'(k_t')^{-1}}$ 和 $KEK_i' = (kek_i')^{\frac{1}{v_q}}$ $(v_q' = \varphi')$。最后,AM 用 $\{\lambda_i, v'_q, kek'_i, KEK'_i\}$ 取代 $\{\lambda_i, v_q, kek_i, KEK_i\}$;且对于 $\lambda_i \in A - \Lambda(1 \leqslant i \leqslant m)$,$t_i' = t_i$,$d_i = g^{rt_i}$。

(2)重加密:密钥更新后,AM 执行与密文相关的重加密操作。

1) AM 随机选择 $h' \in Z_p^*$,执行重加密操作 $C_0' = C_0 e(g,g)^{\alpha h'} = Me(g,g)^{\alpha(h+h')}$,且 $T_t' = T_t^{\frac{1}{\chi_t}}$,$C_{t,j} = (T_t'^{h_j'})^{k_t'}$,$\forall \lambda_{i,j} \in Y \backslash \iota$:$\hat{C}_{i,j} = (T_i^{h_j'})^{k_i}$,其中 $h_j' = h_j + h'$,当 $\lambda_{i,j} \in Y \backslash Y_\vee$ 时,$h_j' = h'$;当 $\lambda_{i,j} \in Y_\vee \cup Y_\wedge$ 时,则新的密文 $\overline{\overline{CT}} = \{\tau, C_0', C_{t,j}, \forall \lambda_{i,j} \in Y \backslash \iota : \hat{C}_{i,j}\}$。

2)密文更新后,AM 生成新的头部信息如下:

$$Hdr' = \{\langle v_q', E(k_t') = g^{\frac{k_t' v_q'}{t_i'}} \rangle_{v_q' \in node(\Phi_t)};\ \forall \lambda_{i,j} \in Y \backslash \iota, \langle v_q, E(k_t) = g^{\frac{k_i v_q}{t_i}} \rangle_{v_q \in node(\Phi_t)}\}$$

当用户要访问数据时,将会收到更新后的 $\{\overline{\overline{CT}}, Hdr'\}$。

5.3 属性增加

(1)私有密钥分发:对每个属性 $\lambda_i \in \Psi(1 \leqslant i \leqslant m)$,TA 随机选择了 $t_i \in Z_p^*$,AM 随机选择了 $\ddot{\chi}_i \in Z_p$,使得 $\ddot{t}_i = t_i \cdot \ddot{\chi}_i$,计算 $D_i = g^{r\ddot{t}_i}$,发送 $\{\lambda_i, D_i = g^{r\ddot{t}_i}\}$ 给用户。

(2)密钥更新:对每个属性 $\lambda_i \in \Psi(1 \leqslant i \leqslant m)$,AM 更新属性用户组为 Φ_q。同时,节点的输出函数 $node()$ 将变成 $node(\Phi_q)$。对任意一个用户 $u_k \in \Phi_q$,AM 执行交操作 $\varphi'' = node(\Phi_q) \cap path(u_k)$,且随机选择 $\ddot{k}_t \in Z_p^*$ $(\ddot{k}_t \neq k_t)$,$k\ddot{e}k_i = g^{\alpha \ddot{t}_i (\ddot{k}_t)^{-1}}$ 和 $K\ddot{E}K_i = (k\ddot{e}k_i)^{\frac{1}{\ddot{v}_q}}$ $(\ddot{v}_q = \varphi'')$,最后 AM 用 $\{\lambda_i, \ddot{v}_q, k\ddot{e}k_i, K\ddot{E}K_i\}$ 取代 $\{\lambda_i, v_q, kek_i, KEK_i\}$,且对于 $\lambda_i \in A - \Psi(1 \leqslant i \leqslant m)$,$\ddot{t}_i = t_i$。

(3)重加密:密钥更新后,AM 执行与密文相关的重加密操作。

1) AM 随机选择 $h' \in Z_p^*$,执行重加密操作 $\overline{C_0} = C_0 e(g,g)^{\alpha h'} = Me(g,g)^{\alpha(h+h')}$,且 $\ddot{T}_i = T_i^{\frac{1}{\ddot{\chi}_i}}$,$C_{t,j} = (\ddot{T}_t^{h'_j})^{\ddot{k}_t}$,$\forall \lambda_{i,j} \in Y \backslash \iota$:$\ddot{C}_{i,j} = (T_i^{h_j'})^{k_i}$,其中 $h_j' = h_j + h'$,当 $\lambda_{i,j} \in Y \backslash Y_\vee$ 时,$h_j' = h'$;当 $\lambda_{i,j} \in Y_\vee \cup Y_\wedge$ 时,新的密文 $\ddot{C}T = \{\tau, \overline{C_0}, C_{t,j}, \forall \lambda_{i,j} \in Y \backslash \iota : \ddot{C}_{i,j}\}$。

2)密文更新后,AM 生成新的头部信息如下:

$$\ddot{H}dr = \{\langle \ddot{v}_q, E(\ddot{k}_t) = g^{\frac{\ddot{k}_i \ddot{v}_q}{t_i}} \rangle_{\ddot{v}_q \in node(\Phi_q)};\ \forall \lambda_{i,j} \in Y \backslash \iota, \langle v_q, E(k_i) = g^{\frac{k_i v_q}{t_i}} \rangle_{v_q \in node(\Phi_t)}\}$$

当用户要访问数据时,将会收到更新后的 $\{\ddot{C}T, \ddot{H}dr\}$。

6 方案分析

本文将文献[14]中的方案与本文方案在空间代价和时间代价这两个方面作了比较,具体比较结果如表 2 所列。

本文支持属性的撤销和增加,考虑了属性变动后,本文方案与文献[14]中的方案在具备同样的安全性条件下,其密文更新的数量会减少,从而降低了管理器的计算开销,并在一定程度上缩短密文的长度,进一步地提高方案的效率。表 2 中,L_0 和 L_1 表示群 G_1 和 G_T 中元素的长度,θ 表示全体用户属性的个数,t 表示与密文相关的属性个数,e 代表双线性对运算,exp 表示指数运算,k 表示撤销或增加属性的个数,m 表示系统中属性的个数,ρ 表示一个用户的属性个数。表 3 中 q 指本文需要更新的密文数量,n 指文献[14]中密文更新的数量。

表 2 列出了文献[14]中的方案与本文方案在计算代价上的比较结果。

表 2 方案比较结果

Table 2 Comparison of different schemes

	文献[14]中的方案	本文方案
初始化	$(m+2)\exp+e$	$m\exp+e$
密钥生成	$(3m+2)\exp$	$(2m+1)\exp$
KEK 生成	$m\exp$	$m\exp$
加密算法	$(2m+1)\exp+me$	$m\exp+me$
属性组密钥解密		$2\rho\exp$
私钥长度	$(2\theta+1)L_0$	$(\theta+1)L_0$
密文长度	$(3t+1)L_0+L_1$	$2tL_0+L_1$
公钥长度	$4L_0+L_1$	L_0+L_1

表 3 列出了文献[14]中的方案与本文方案实施一次属性变动原子操作时在密文更新数量上的比较结果。

表 3 密文更新分析表

Table 3 Ciphertext update analysis

		文献[14]中的方案	本文方案
用户的数量	撤销或增加属性的数量	密文更新的数量	密文更新的数量
1	1	$3n\exp+ne$	$2q\exp+qe$
1	i	$3ni\exp+nie$	$2qi\exp+qie$
σ	1	$3n\sigma\exp+n\sigma e$	$2q\sigma\exp+q\sigma e$
σ	i	$3ni\sigma\exp+ni\sigma e$	$2qi\sigma\exp+qi\sigma e$

7 安全性分析

定理 1 若 CDH 假设成立,则本文方案在选择明文攻击模型下是安全的。

证明:挑战者生成双线性群 $\{p, G_1, G_T, e, g\}$,G_1 和 G_T 是素数阶 p 的循环群,g 是 G_1 的生成元。挑战者随机选择 x_1,$x_2 \in Z_p$,并且计算 $\bar{A} = g^{x_1}$,$B = g^{x_2}$,将 $\bar{A}$,B 发送给模拟器。

系统初始化过程如下:敌手选择了目标访问树 τ^* 和挑战的属性 λ^*,并将它们发送给模拟器。λ^* 是满足 τ^* 的必要属性之一。

管理器初始化过程如下:算法模拟了系统的公共参数和属性管理器的公钥。

1)模拟器挑选了 $\alpha \in Z_p$,计算 $y = e(g,g)^\alpha$,公共参数

$PP=\{g,e(g,g)^{\alpha}\}$和主密钥 $MK=\alpha$。

2)对每个属性 $\lambda_i\in A-(\lambda^*)$，模拟器选择 $t_i\in Z_p^*$，计算 $T_i=g^{t_i^{-1}}$，若 $\lambda_i=\lambda^*$，则 $T^*=g_{t^*}^{\frac{1}{t^*}}$，其中 $t^*\in Z_p^*$。

因此，AM 的公钥是 $MPK=\{T_i|\lambda_i\in A-(\lambda^*)\}\cup\{T^*\}$。AM 的主密钥 $MMK=\{t_i|\lambda_i\in A-(\lambda^*)\}\cup\{t^*\}$。然后模拟器更新 λ^* 密文 $\overline{T^*}=T^{*x_1}=\bar{A}_{t^*}^{\frac{1}{t^*}}$，设置 $\overline{t^*}=x_1t^*$。模拟器更新管理器 $\overline{MPK}=\{T_i|1\leqslant i\leqslant m,\lambda_i\neq\lambda^*\}\cup\{\overline{T^*}\}$ 和管理器主密钥 $\overline{MMK}=\{t_i|1\leqslant i\leqslant m,\lambda_i\neq\lambda^*\}\cup\{\overline{t}^*\}$。

阶段 1　阶段 1 的挑战如下：在这个过程中，模拟器保留两个列表，分别为 L_1 和 L_2，初始情况下这两个列表为空。

类型 1　私钥询问$\langle u_1,\omega_{u_1}\rangle$，$\omega_{u_1}$ 满足 τ^*，但是 u_1 的属性 λ^* 已被撤销，模拟器首先检查 u_1 是否出现在 L_1。如$\langle u_1,r_1,r_1^*,KEK_1,DSK_1\rangle$，若 u_1 出现在 L_1，则模拟器直接将 KEK_1 和 DSK_1 发送给挑战者，否则模拟器按照如下方法计算私钥。

(1)模拟器选择一个随机 $r_1\in Z_p$，对每个属性 $\lambda_i\in\omega_{u_1}-(\lambda^*)(1\leqslant i\leqslant|L_1|)$，$|L_1|$ 是 ω_{u_1} 的大小。模拟器随机选择 $r^*\in Z_p$，相对于 λ^*，$DSK_1=\{D_0=g^{\alpha-r_1},\forall\lambda_i\in\omega_{u_1}-(\lambda^*),D_i=g^{r_1t_i},D_1^*=g^{r_1^*t^*}=B^{r^*t^*}\}$其中包含了设置 $r_1^*=x_2r^*$。

(2)对每个属性 $\lambda_i\in\omega_{u_1}-(\lambda^*)$，随机选择 $k_i\in Z_p^*$，模拟器生成 $\eta_i=node(\Phi_i)\cap path(u_1)$，计算 $kek_i=g^{\alpha t_ik_i^{-1}}$ 和 $KEK_i=(kek_i)^{\frac{1}{v_q}}=g^{\frac{\alpha t_ik_i^{-1}}{v_q}}(v_q\in\eta_i)$；对于 λ^*，随机选择 $k^*\in Z_p^*$ 模拟器以及 $v^*\in path(u_1)$，并且计算 $kek_1^*=g^{\alpha t^*(k^*)^{-1}}$ 和 $KEK_1^*=(kek_1^*)^{\frac{1}{v^*}}$。属性组密钥是 $KEK_1=\{\forall\lambda_i\in\omega_{u_1}-\lambda^*:v_q,kek_i,KEK_i\}\cup\{\lambda^*:v^*,kek_1^*,KEK_1^*\}$。

(3)最后，模拟器增加$\langle u_1,r_1,r_1^*,KEK_1,DSK_1\rangle$到 L_1，发送 DSK_1 和 KEK_1 给敌手。

类型 2　私钥询问$\langle u_2,\omega_{u_2}\rangle$，$\omega_{u_2}$ 不满足 τ^*，但是 u_2 的属性 λ^* 在 ω_{u_2} 中，模拟器首先检查 u_2 是否出现在 L_2。如$\langle u_2,r_2,r_2^*,KEK_2,DSK_2\rangle$，若 u_2 出现在 L_2，模拟器则直接将 KEK_2 和 DSK_2 发送给挑战者，否则将按照如下方式计算私钥。

(1)模拟器选择一个随机 $r_2\in Z_p$，对于每个属性 $\lambda_i\in\omega_{u_2}-(\lambda^*)(1\leqslant i\leqslant|L_2|)$，$|L_2|$ 是 ω_{u_2} 的大小。模拟器随机选择 $r^*\in Z_p$，相对于 λ^*，$DSK_2=\{D_0=g^{\alpha-r_2},\forall\lambda_i\in\omega_{u_2}-(\lambda^*),D_i=g^{r_2t_i},\lambda_i=\lambda^*:D_2^*=g^{r_2^*t^*}\}$。

(2)对每个属性 $\lambda_i\in\omega_{u_2}-(\lambda^*)$，随机选择 $k_i\in Z_p^*$，模拟器生成 $\eta_i=node(\Phi_i)\cap path(u_2)$，计算 $kek_2=g^{\alpha t_ik_i^{-1}}$ 和 $KEK_i=(kek_i)^{\frac{1}{v_q}}$；对于 λ^*，模拟器随机选择 $k^*\in Z_p^*$ 以及 $v^*\in path(u_2)$，并且计算 $kek_2^*=g^{\alpha x_1t^*(k^*)^{-1}}=\bar{A}^{\alpha t^*(k^*)^{-1}}$ 和 $KEK_2^*=(kek_2^*)^{\frac{1}{v^*}}$。属性组密钥是 $KEK_2=\{\forall\lambda_i\in\omega_{u_2}-\lambda^*:v_q,kek_i,KEK_i\}\cup\{\lambda^*:v^*,kek_2^*,KEK_2^*\}$。

(3)最后，模拟器增加$\langle u_2,r_2,r_2^*,KEK_2,DSK_2\rangle$到 L_2，发送 DSK_2 和 KEK_2 给敌手。

挑战：敌手确定阶段 1 结束，它发送两个等长信息 M_0 和 M_1，模拟器随机抛掷一枚硬币 $\mu\in\{0,1\}$，并计算挑战密文如下：

(1)第一层加密：计算 $C_0=M_\mu e(g,g)^{\alpha h}$。

(2)第二层加密：设 τ^* 的根节点的值为 h，标记未分配所有子节点和已分配的根节点，对每一个未分配的非叶子节点执行以下步骤。

若符号为 $\wedge$，标记它的子节点为未分配，通过模加机制给子节点赋值，除最后一个子节点外其他每个子节点赋予一个随机值 $h_j(1\leqslant h_j\leqslant p-1)$，其中最后一个孩子节点的赋值为 $h_\zeta=h-\sum_{j=1}^{\zeta-1}h_j \bmod p$，标记这个节点为已赋值；若符号为 $\vee$，则将每个子节点赋值为 h，并标记该节点为已赋值。

(3)对于每一个叶子属性 $\lambda_{i,j}\in Y$，计算 $C_{i,j}=T_i^{h_j}$，对于 $\lambda^*\neq\lambda_{i,j}$，且 $\lambda^*\in\tau^*$，模拟器随机选择 $k_i\in Z_p^*$，并计算 $C_{i,j}^*=T_i^{h_jk_i}$；对于 $\lambda^*=\lambda_{i,j}$，计算 $C^*=T^{*h_jk^*}=\bar{A}^{h_jk^*}$，其中暗含着 $k^*=x_1k_i$，模拟器最终得到的密文为 $CT^*=\{C_0,C_{i,j}^*,C^*\}$，对于 $\lambda_i\in\tau^*$，模拟器使用 $node(\Phi_i)$算法，计算头部信息：

$$Hdr^*=\{\langle v_q,E(k_i)=g^{\frac{k_iv_q}{t_i}}\rangle_{v_q\in node(\Phi_i)}\}$$

$$Hdr^*=\{\langle v_q^*,E(k^*)=g^{\frac{k^*v_q^*}{t^*}}\rangle_{v_q^*\in node(\Phi^*)}\}$$

最后，模拟器将$\{CT_\mu^*,Hdr^*\}$发送给挑战者。

阶段 2　阶段 2 类似询问阶段 1。

猜测：最后，模拟器忽略挑战者的输出，随机从 L_1,L_2 中选择$\langle u_1,r_1,r_1^*,KEK_1,DSK_1\rangle$和$\langle u_2,r_2,r_2^*,KEK_2,DSK_2\rangle$。

如果敌手解密可能的挑战密文，则必须询问所包含的两部分密钥。在 $\frac{e(D_0,g)e(D_1^*,T^*)}{e(KEK_2^*,E(k^*))}$ 这个式子中，使得 $e(D_0,g)e(D_1^*,T^*)=e(KEK_2^*,E(k^*))$ 成立，即 $B^{r_1^*}=g^{x_1x_2r_2}$，模拟器可以计算得到 $KEK_2^*=g^{\frac{\alpha t^*(k^*)^{-1}}{v^*}}$，并将其作为结果。

假设挑战者进行了 q_1 次 $Type_1$ 询问和 q_2 次 $Type_2$ 询问，模拟器选择出正确的 KEK_1 和 DSK_2 的可能性是$\frac{1}{q_1q_2}$，这说明了模拟器攻破的优势至多为$\frac{\varepsilon}{q_1q_2}$。

抗合谋性分析：本文方案中用户的密钥分为两部分：私有密钥 DSK 和密钥加密密钥 KEK。假设某一个撤销属性 λ_i 的用户 u 的 DSK，与未发生属性撤销的其他用户 u 的 KEK 相结合，此时撤销该属性的用户不能用现有密钥中关于 λ_i 的成分对相关密文进行解密，即发生属性撤销用户的私有密钥部分 $d_i=g^{r_i}$ 与现有用户的加密密钥部分 $kek_i'=g^{\alpha t_i\chi_ik_i'^{-1}}$ 结合，但未能够得到完整的解密密钥。因为 χ_i,k_i 及 k_i'是随机选择的，所以 $\prod_{\lambda_i\in\omega_{u_a}}e(kek_i',\hat{C}_{i,j})=\prod_{\lambda_i\in\omega_{u_a}}e(g^{\alpha t_i\chi_ik_i'^{-1}},T_i'^{h_jk_i})\neq e(g,g)^{\alpha h}$。因此发生属性撤销后的用户在解密时无法得到所需要的 $e(g,g)^{\alpha h}$，故此方案在抗合谋攻击方面是安全的。

8　实验分析

为了验证方案的正确性和效率，本文对实现方案中的各种算法进行了测试，实验测试环境如表 4 所列。首先本文用 20 个属性构成的访问策略分别对文献[14]中的方案和本文方案进行比较，两个方案各执行了 50 次，最终各自取其平均

值作为最后的结果。两个方案的对比如图 7 所示。

表 4 实验环境

Table 4 Experimental environment

操作系统	Windows10(64bit)家庭中文版
系统运行环境	Inter(R)Core(TM) i3-10110v CPU @ 2.10GHz 的 PC 机，内存为 4GB
软件环境	JDK17
算法	Java 密码库 JPBC

从图 7 可以得出结论：本文方案在初始化算法、密钥生成及解密算法的执行时间上更短，降低了 AM 的计算量，但在加密算法上运行的时间长于文献[14]的加密时间，这是因为本文在加密算法上利用模加机制对与节点的每个子节点进行分配，给或节点的每一个子节点直接赋值为秘密值，而文献[14]采用的树型访问结构用一个多项式为每个节点赋值。

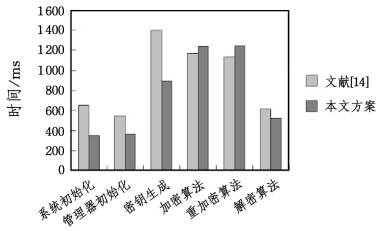

图 7 算法运行对比表

Fig. 7 Comparison of algorithm operation

结束语 针对 CP-ABE 方案中属性撤销和添加的问题，本文提出了一种基于懒惰模式密文更新的属性变动优化方案，利用属性密文组和用户密文组对密文更新进行了分析，即通过某个撤销(增加)属性的属性密文集合与撤销属性前(增加属性后)的用户密文组做交集，缩小需要更新的密文范围并减少密文更新的次数，提高了本文方案的有效性。安全证明表明本文方案在选择明文攻击下是安全的。未来研究的工作重点是在属性变动的多次原子操作执行过程中如何避免不必要的密文更新，使得方案更加有效。

参 考 文 献

[1] BETHENCOURT J, SAHAL A, WATERS B. Ciphertext-Policy Attribute-Based Encryption[C]// IEEE Symposium on Security & Privacy. IEEE Computer Society, 2007: 321-334.

[2] GOYAL V, PANDEY O, SAHAI A, et al. Attribute-based Encryption for Fine-grained Access Control of Encrypted Data [C]// Proceedings of the 13th ACM Conference on Computer and Communications Security. USA, 2006: 89-98.

[3] WATERS B. Ciphertext-policy Attribute-based Encryption: an Expressive, Efficient, and Provably Secure Realization[C]// International Workshop on Public Key Cryptography. Berlin: Springer, 2008: 53-70.

[4] ZU L, LIU Z, LI J. New Ciphertext-policy Attribute-based Encryption with Efficient Revocation[C]// 2014 IEEE International Conference on Computer and Information Technology (CIT). IEEE, 2014: 281-287.

[5] XIE X, MA H, LI J, et al. An Efficient Ciphertext-Policy Attribute-Based Access Control Towards Revocation in Cloud Computing [J]. Journal of Universal Computer Science, 2013, 19(16): 2349-2367.

[6] HUR J, DONG K N. Attribute-Based Access Control with Efficient Revocation in Data Outsourcing Systems[J]. IEEE Transactions on Parallel & Distributed Systems, 2011, 22(7): 1214-1221.

[7] LIU C W, HSIEN W F, YANG C C, et al. A Survey of Attribute-Based Access Control with User Revocation in Cloud Data Storage[J]. International Journal of Network Security, 2016, 18(5): 900-916.

[8] YONG C, WANG Z Y, MA J, et al. Efficient Revocation in Ciphertext-Policy Attribute-Based Encryption Based Cryptographic Cloud Storage[J]. Journal of Zhejiang University-SCIENCE C(Computers & Electronics), 2013, 14(2): 85-97.

[9] WANG P P, FENG D G, ZHANG L W. CP-ABE scheme supporting fully fine-grained attribute revocation [J]. Journal of Software, 2012, 23(10): 2805-2816.

[10] PIRRETTI M, TRAYNOR P, MCDANIEL P, et al. Secure Attribute-Based Systems[J]. Journal of Computer Security, 2010, 18(5): 799-837.

[11] FH A, MWA B, ST A, et al. A Revocable and Outsourced Multi-Authority Attribute-Based Encryption Scheme in Fog Computing[J]. Computer Networks, 2021(10): 1-8.

[12] LI J, SHI Y, ZHANG Y. Searchable Ciphertext-Policy Attribute-Based Encryption with Revo-cation in Cloud Storage[J]. International Journal of Communication Systems, 2017, 30(1): 2933-2947.

[13] ZHANG W F, CHEN Z, LIU X D, et al. CP-ABE scheme supporting Fine-grained attribute direct revocation[J]. Journal of Software, 2019, 30(9): 2760-2771.

[14] LI J, YAO W, HAN J, et al. User Collusion Avoidance CP-ABE with Efficient Attribute Revocation for Cloud Storage[J]. IEEE Systems Journal, 2018(12): 1767-1777.

[15] SUN L, ZHAO Z Y, WANG J H, et al. Attribute-based encryption scheme supporting attribute revocation in cloud storage environment[J]. Journal of Communications, 2019, 40(5): 47-56.

[16] YAN X X, TANG Y L. Attribute-based encryption scheme with efficient revocation in data outsourcing systems[J]. Journal on Communications, 2015, 36(10): 92-100.

LEI Xue-jiao, born in 1997, postgraduate. Her main research interests include information security and cryptography.

Nurmamat HELIL, born in 1976, Ph.D, professor, Ph. D supervisor. His main research interests include information system security, access control, and cloud storage security.

(责任编辑：喻藜)