计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2017.20151135
54(7):1603-1616,2017

支持多用户协同编辑的云存储访问控制方法

史姣丽[1,2] 黄传河[1] 何 凯[1] 沈燮阳[1] 华 超[1]
[1](武汉大学计算机学院 武汉 430072)
[2](九江学院 江西九江 332005)
(shijiaoli@whu.edu.cn)

An Access Control Method Supporting Multi-User Collaborative Edit in Cloud Storage

Shi Jiaoli[1,2], Huang Chuanhe[1], He Kai[1], Shen Xieyang[1], and Hua Chao[1]
[1](*School of Computer Science, Wuhan University, Wuhan* 430072)
[2](*Jiujiang University, Jiujiang, Jiangxi* 332005)

Abstract As for attribute-based access control in cloud storage, most of researches focus on reading permission control when multiple users read the same out-sourced data simultaneously. They dot't consider writing permission control when multiple users modify the same data simultaneously. In multi-user collaborative edit scene, challenges have emerged: 1) A data owner with limited capabilities of computation, storage and communication, would like cloud to aid him with writing permission control, but would not like it to know the content of data, or get what is matched, or even predict the users' writing permission either. 2) Boolean formula cannot describe writing permission policy. 3) Bilinear pairing operations bring great computational costs. In this work, a collaborative edit access control method is presented in cloud storage. That is, a data owner defines writing permission policy represented by a circuit, and semi-trusted cloud decides whether or not the writing succeeds by matching writing policy without the prediction of acceptability of the next edit request. Analyses and simulations show that our method is provided with the ability of multi-user collaborative access control for cloud storage, and the storage cost and the computation cost of encrypting and decrypting are both lesser at user end in reading permission control with cloud-aided decryption.

Key words cloud storage; access control; attribute-based encryption (ABE); multi-user collaborative edit; cloud-aided writing permission control

摘 要 以往的云存储属性基访问控制研究大多数是对外包数据的读权限进行控制,很少考虑多个用户协同编辑云端同一个外包数据时的写权限控制.多用户协同编辑控制存在挑战:1)资源有限的数据拥有者希望云辅助自己对外包数据的写权限进行控制,但不希望云获得数据内容,也不希望云感知匹配内容,甚至不希望云能够预测用户的写权限;2)布尔公式的访问结构无法表达写权限控制策略;3)双线性映射运算计算代价大,不适合多用户协同编辑控制.提出一个支持多用户协同编辑的云存储访问控制方法:数据拥有者采用表达能力更丰富的 circuit 定义写权限访问控制策略,委托半可信云采用矩阵运算快速判断用户提交的修改数据是否应该接受,并且云不可预测每个用户是否具有写权限.分析与实验表

收稿日期:2015-12-20;**修回日期**:2016-10-10
基金项目:国家自然科学基金项目(61373040,61572370)
This work was supported by the National Natural Science Foundation of China (61373040, 61572370).
通信作者:黄传河(huangch@whu.edu.cn)

明:该方法具有多用户协同编辑的访问控制能力,并且在读权限控制时,利用云辅助解密方法使得用户端存储量和加解密计算量是较小的.

关键词 云存储;访问控制;属性加密;多用户协同编辑;云辅助写权限控制

中图法分类号 TP309.2; TP303

云作为近年来发展较快的分布式架构,不仅具有存储数据的能力,而且能为大量用户提供随时随地的访问.但是,云服务器不能完全被用户信任,因而需要研究合适的云存储访问控制方法:1)控制合法用户能够访问授权数据;2)控制半可信的云服务器无法得知数据的内容.

属性加密(attribute-based encryption, ABE)访问控制方法减弱了数据拥有者和数据访问者之间的耦合性,因而比传统访问控制方法更适合上述云存储访问控制应用场景.ABE分为2类: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).前者适应于用户检索外包数据的应用场景;后者适应于用户读取外包数据的应用场景.后者(CP-ABE)采用Data-binding-policy方法(将访问策略绑定在数据上)保证外包数据可以被多个用户按需读取,因而企业无需建设数据基础设施.然而,大多数研究只关注用户对云端密文的读权限控制[1-2],很少有研究关注上述无基础设施的情况下如何实现密文协同编辑控制.

多个用户同时编辑同一个外包密文数据时,可能出现5种控制要求:

1) 多个用户的写权限级别不同,他们同时提交修改数据时,需要保留高权限用户提交的修改数据,忽略低权限用户提交的修改数据;

2) 恶意编辑者一直写数据,致使正常的其他编辑者无法写成功;

3) 云环境下多个用户提交的修改数据在传输时时延不同,会使得修改数据的发出时间顺序与到达服务器的时间顺序不同,数据在具体应用时需要确定保留修改数据版本的时间顺序;

4) 每个用户的多个访问状态之间存在偏序关系,例如用户 A 在某个状态提出访问数据请求时,要求该用户曾经经历过其他状态;

5) 多个用户之间在访问状态上存在约束关系,例如用户 A 在某个状态提出数据请求时,要求其他用户集合$\{u_j\}$的状态满足指定的偏序关系.

以往的云存储访问控制在处理多用户协同编辑时,都是按照修改数据的到达时间(如文献[3]),如果直接使用以前研究所提出的任何一种云存储访问控制方法,都无法实现上述多用户协同编辑细粒度访问控制需求.

协同编辑访问控制与读权限访问控制存在不同:读访问时,决定是否可读的是用户端的属性私钥,即如果用户的属性私钥满足密文关联的访问策略,那么该用户就可以读取密文.协同编辑时,由于数据拥有者的终端设备在计算、存储和通信等方面能力有限,因而需要云辅助数据拥有者实现写权限控制.然而,在云辅助写权限控制时存在2种挑战:1)数据拥有者希望云辅助自己判断是否接受每个用户的修改数据;2)数据拥有者又不希望云获得数据内容,更不希望云能预测用户下一次提交的数据是否可写.

考虑以上安全控制需求,本文尝试在云存储环境下研究多用户协调编辑访问控制方法,主要贡献总结有3点:

1) 设计了云辅助解密方法,减轻了用户端的计算代价,并证明了该方法在离散对数困难问题假设下没有减弱安全性.

2) 提出了多用户协同编辑访问控制方法.该方法支持Collaborative-Edit:考虑到常用的布尔公式访问结构无法表达复杂的写权限控制策略,Owner采用表达能力更丰富的circuit定义复杂的写权限控制策略,将策略关联到密文数据,然后上传到云.多个用户提出编辑请求时,云根据Owner定义的策略确定最新数据版本,实现Data-binding-policy的细粒度写权限控制.

3) 设计了MOR(many-to-one recoding)原语,对TOR(two-to-one recoding)原语[4]进行维度扩展,基于LWE(learning with error)理论实例化所设计的MOR原语,实现了多用户协同编辑访问控制方法:考虑到基于双线性映射及其安全假设构建的ABE访问控制方法计算代价大,本文采用代价更小的矩阵运算实现写策略匹配,能够在半信任云上快速判断用户提交的修改数据是否应该接受,并且云不可预测每个用户是否具有写权限.

1 相关工作

支持协同编辑的云存储访问控制研究方面,Dong 等人[3]基于 HIBE(hierarchical identity-based encryption)设计了 SECO 方案,该方案是根据修改数据到达云服务器的时间来确定最终的数据版本;Ruj 等人[5]只提出了使用 Claim Policy 控制单个用户写权限控制;Li 等人[6]将时间划分为时间片,用 Hash 链和签名技术控制写权限,但只考虑 Create-Writing 情形,没有考虑 Collaborative-Writing 情形;范艳芳等人[7]考虑协作环境下的访问控制模型,在经典 BLP 模型上扩充主客体的安全标记,提高了传统访问控制模型的灵活性.

ABE 方案的研究方面,Yang 等人[1]考虑大数据访问控制时的动态策略更新问题,讨论了布尔公式、门限结构及 LSSS 结构的更新过程;Lai 等人[8]使用半信任代理,在密文上修改访问策略,同时不泄漏任何信息给半信任代理;Chun 等人[9]提出了支持动态组成员管理的 ABE 方案;Hur 等人[2]在军用容断网络的应用环境中,设计的 CP-ABE 方案支持多个 AA 分发重复属性;Rouselakis 等人[10]指出应该由 Owner 自己指定其数据的访问属性;王皓等人[11]给出了外包 ABE 的形式化定义,构造并证明了一个外包 ABE 的方案;Li 等人[6]针对 PHR (personal health records)系统的访问控制需求,将属性域分为个人属性域和公共属性域,个人属性域由每个 Owner 自己进行定义,而公共属性域由 AA 进行定义,ABE 方案的格实现方面,研究者认为格可以抵御量子计算机的攻击;Boyen[12]提出了适合处理复杂访问策略的格操作框架:私钥用 Ephemeral Lattices,论文利用 Basis Splicing 技术,允许接受者将 Ephemeral Lattices 基转换为任意其他格的基,可以处理规范或者非规范的访问策略;Liu 等人[13]提出了基于格的门限层次 ABE(t-HABE),在 LWE 困难问题假设下,降低了属性数量;ABE 方案的高效性研究方面,Hohenberger 等人[14]将 ABE 方案分成线上和线下 2 部分,减少了线上计算量.

在实现 ABE 方案时,访问策略的形式可以是布尔公式、circuit 等. Garg 等人[15]基于多线性映射构造了一个 circuit ABE 方案:只加密 1 位明文,密文长度只与输入集合中 $x_i=1$ 的个数有关,用户私钥与 circuit 的深度 depth 有关,公钥与 input 的数量 n 有关;Xu 等人[16]在 Garg 等人[15]的基础上,设计了 Complement circuit CP-ABE,实现了云辅助解密的结果验证,可以防止云服务器欺骗用户.

circuit ABE 方案也可以基于 LWE 困难假设构造,其理论基础是 Regev[17]证明的结论:解决 LWE 平均困难问题相当于最坏困难问题,因此,LWE 困难假设可以作为密码学的一个 Central Fixture;Gorbunov 等人[4]设计了一个 TOR 原语,并基于 LWE 困难假设实例化了 TOR 原语,提出了一个 circuit ABE 方案:策略是以扇入度为 2 的 Gates 组成的逻辑电路,且私钥的长度与 circuit 的长度成正比.

2 MOR 原语及实例化设计

2.1 MOR 原语

存在 1 组伪随机函数 $Encode(\cdot)$ 和 1 个重编码密钥 rk,使得转换可以实施:

$$(Encode(\boldsymbol{pk}_1,w),Encode(\boldsymbol{pk}_2,w),\cdots,Encode(\boldsymbol{pk}_N,w))\mapsto Encode(\boldsymbol{pk}_{tgt},w),$$

转换密钥 rk 可以由任意公钥 $\boldsymbol{pk}_b$ 对应的私钥 $\boldsymbol{sk}_b$ 生成,w 是高斯种子随机数.

2.2 设计思路

在 Gorbunov 等人[4]基于 LWE 困难假设实例化的 TOR 原语中,有:

$$\boldsymbol{R}^{\mathrm{T}}\begin{pmatrix}\boldsymbol{A}_0^{\mathrm{T}}\boldsymbol{s}+\boldsymbol{e}_0\\ \boldsymbol{A}_1^{\mathrm{T}}\boldsymbol{s}+\boldsymbol{e}_1\end{pmatrix}\approx\boldsymbol{A}_{tgt}^{\mathrm{T}}\boldsymbol{s},\tag{1}$$

其中,$\boldsymbol{R}\in\mathbb{Z}_q^{2m\times m}$,$\boldsymbol{A}\in\mathbb{Z}_q^{n\times m}$,$\boldsymbol{s}\in\mathbb{Z}_q^n$,$\boldsymbol{R}$ 是满足$[\boldsymbol{A}_0\|\boldsymbol{A}_1]\boldsymbol{R}=\boldsymbol{A}_{tgt}$的一个低秩矩阵,映射为密钥 rk;矩阵 $\boldsymbol{A}_b\in\mathbb{Z}_q^{n\times m}$($\boldsymbol{A}_0$ 或 $\boldsymbol{A}_1$)映射为公钥 $\boldsymbol{pk}_b$($\boldsymbol{pk}_0$ 或 $\boldsymbol{pk}_1$),后门矩阵 $\boldsymbol{T}_b\in\mathbb{Z}_q^{m\times m}$($\boldsymbol{T}_0$ 或 $\boldsymbol{T}_1$)映射为私钥 $\boldsymbol{sk}_b$($\boldsymbol{sk}_0$ 或 $\boldsymbol{sk}_1$),$\boldsymbol{A}_b^{\mathrm{T}}\boldsymbol{s}+\boldsymbol{e}_b$ 映射为编码过程 $Encode(\boldsymbol{A}_b,w)$. $\boldsymbol{e}_b$ 满足高斯分布 $D_{\mathbb{Z}^m,\sigma}$,界为 $\sqrt{m}\,\sigma$. 矩阵 $\boldsymbol{R}^{\mathrm{T}}$ 表示对矩阵 $\boldsymbol{R}$ 进行转置运算.

根据 LWE 困难假设[17],能够区分$\{\boldsymbol{A},\boldsymbol{A}^{\mathrm{T}}\boldsymbol{s}+\boldsymbol{e}\}$和$\{\boldsymbol{A},\boldsymbol{u}\}$两个分布的优势可以忽略,其中,$\boldsymbol{u}\in\mathbb{Z}_q^m$. 因而,式(1)可以扩展为

$$\boldsymbol{R}^{\mathrm{T}}\begin{pmatrix}\boldsymbol{A}_1^{\mathrm{T}}\boldsymbol{s}+\boldsymbol{e}_1\\ \boldsymbol{A}_2^{\mathrm{T}}\boldsymbol{s}+\boldsymbol{e}_2\\ \vdots\\ \boldsymbol{A}_N^{\mathrm{T}}\boldsymbol{s}+\boldsymbol{e}_N\end{pmatrix}\approx\boldsymbol{A}_{tgt}^{\mathrm{T}}\boldsymbol{s},\tag{2}$$

其中,$\boldsymbol{R}\in\mathbb{Z}_q^{Nm\times m}$.

2.3 实例化设计

根据式(2),对 MOR 原语进行详细的实例化设计.

$Params(1^{\lambda}, d_{\max})$:选择 LWE 的相关参数,输出全局公共参数 $PP=(n, X, q, m, w)$,其中,$n=n(\lambda)$ 是 LWE 的维度,$X=X(\lambda)=D_{\mathbf{Z},\sqrt{n}}$ 是 Error 的分布,$B=B(n)=O(n)$ 是 Error 的界,$d_{\max}$ 是 $Recode(\cdot)$ 算法运行次数的界,$q=q(n)=\tilde{Q}(n^2 d_{\max})^{d_{\max}} n$ 是 LWE 的模,$m=m(n)=O(n\log q)$ 是抽样数量,$w=w(n)=O(\sqrt{n\log q})$ 是高斯参数.

$Keygen(PP)$:调用后门函数,生成矩阵 $\boldsymbol{A}$ 及其后门矩阵 $\boldsymbol{T}$,分别作为公钥和私钥:$\boldsymbol{pk}=\boldsymbol{A}$,$\boldsymbol{sk}=\boldsymbol{T}$,其中,$\boldsymbol{A}\in\mathbb{Z}_q^{n\times m}$,$T\in\mathbb{Z}_q^{m\times m}$.

$Encode(\boldsymbol{pk}, w)$:选择 Error 向量 $\boldsymbol{e}\in X^m$ 和 $\boldsymbol{s}\in\mathbb{Z}_q^n$,计算并输出 $\boldsymbol{\varphi}=\boldsymbol{A}^{\mathrm{T}}\boldsymbol{s}+\boldsymbol{e}$.

$ReKeygen(\boldsymbol{pk}_1, \boldsymbol{pk}_2, \cdots, \boldsymbol{pk}_N, \boldsymbol{sk}_b, \boldsymbol{pk}_{\mathrm{tgt}})$:选择 $N-1$ 个离散高斯矩阵 $\boldsymbol{R}_i\in(D_{\mathbf{Z},\sigma})^{m\times m}$,计算 $\boldsymbol{U}=\boldsymbol{pk}_{\mathrm{tgt}}-\sum_{i=1}^{n}\boldsymbol{A}_i\boldsymbol{R}_i$,其中,$i\neq b$,计算 $\boldsymbol{R}_b=SampleD(\boldsymbol{A}_b, \boldsymbol{T}_b, \boldsymbol{U})$,输出:

$$rk^{\mathrm{tgt}}=\begin{bmatrix}\boldsymbol{R}_1\\ \boldsymbol{R}_2\\ \vdots\\ \boldsymbol{R}_N\end{bmatrix}\in\mathbb{Z}^{Nm\times m}.$$

$Recode(rk^{\mathrm{tgt}}, \boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2, \cdots, \boldsymbol{\varphi}_N)$:计算 $\boldsymbol{\varphi}_{\mathrm{tgt}}=\lceil\boldsymbol{\varphi}_1^{\mathrm{T}}\|\boldsymbol{\varphi}_2^{\mathrm{T}}\|\cdots\|\boldsymbol{\varphi}_N^{\mathrm{T}}\rfloor rk^{\mathrm{tgt}}$.

$E(\boldsymbol{\varphi}, \boldsymbol{\mu})$:加密明文 $\boldsymbol{\mu}$,计算 $\Upsilon=\boldsymbol{\varphi}+\left\lceil\frac{q}{2}\right\rfloor\mu(\mathrm{mod}q)$.

$D(\boldsymbol{\varphi}', \Upsilon)$:解密密文 Υ,计算 $\mu=(Round(\Upsilon_1-\boldsymbol{\varphi}'_1), Round(\Upsilon_2-\boldsymbol{\varphi}'_2), \cdots, Round(\Upsilon_n-\boldsymbol{\varphi}'_n))$,其中:

$$Round(x)=\begin{cases}0, & |x|<q/4.\\ 1, & \text{otherwise.}\end{cases}$$

2.4 MOR 原语的特性

MOR 原语具有不可预测性和可重用性.

1) 不可预测性. 对于 N 输入的门 $G:\{0,1\}\times\cdots\times\{0,1\}\to\{0,1\}$ 和 N 个输入 $L_1, L_2, \cdots, L_N$,已知转换表 $(rk, \boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2, \cdots, \boldsymbol{\varphi}_N, \boldsymbol{\varphi}_{\mathrm{tgt}})$ 和 $\boldsymbol{\varphi}_i=Encode(\boldsymbol{pk}_i, w)$,容易计算 $Encode(G(L_1, L_2, \cdots, L_N), w)$,但 $Encode(1-G(L_1, L_2, \cdots, L_N), w)$ 仍然保持隐藏.

2) 可重用性. $\boldsymbol{s}$ 与转换过程完全独立,因而可以通过选择随机数 $\boldsymbol{s}$ 刷新转换表的方式达到可重用的目的.

3 建 模

3.1 系统模型

如图 1 所示,系统由 4 部分实体组成:属性授权中心(attribute authorities, AA)、云服务器(Cloud Server)、数据拥有者(Owner)、用户(User).

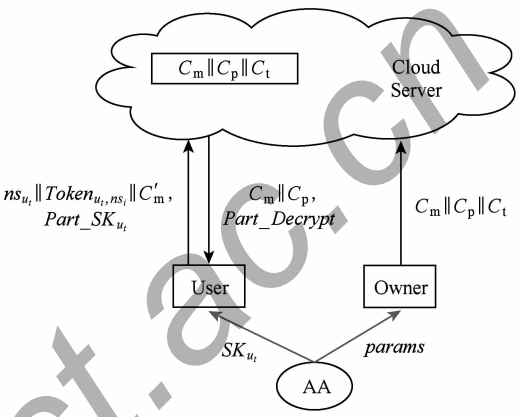


Fig. 1 System model

图 1 系统模型

1) AA 是可信属性授权中心. 负责管理属性集合,为 User 分发代表读权限的私钥 SK_{u_t} 和代表写权限的凭证 $ns_{u_t}\|Token_{u_t,ns_i}$,其中,$u_t$ 表示用户编号,ns_{u_t} 表示用户 u_t 的当前状态,ns_i 表示所有用户的状态向量.

2) Owner 是数据拥有者. Owner 先用内容密钥运行对称密码算法加密,得到 C_m;然后用本地指定的读权限访问策略对内容密钥进行 CP-ABE 加密,得到 C_p;接着定义多用户协同编辑策略 C_t;最后将 $C_m\|C_p\|C_t$ 上传云端.

3) User 是数据的读者和编辑者. 为降低解密计算代价,User 向 Server 提交委托解密密钥 $Part\_SK_{u_t}$,云辅助 User 解密之后,向 User 发送解密中间结果 $Part\_Decrypt$ 和密文 $C_m\|C_p$,然后,User 用自己的属性私钥与密文中的读权限访问策略 C_p 尝试匹配,如果匹配通过,就能读取明文数据. 另外,User 可以在本地修改数据明文,并使用加密时的内容密钥进行对称加密,得到编辑的结果 C'_m,然后将 C'_m 和写权限凭证 $ns_{u_t}\|Token_{u_t,ns_i}$ 上传到 Server.

4) Server 是多云存储环境下的半可信云服务器. Server 永远在线,为 User 提供随时随地的数据访问服务,包括辅助 Owner 进行协同编辑权限匹配. 当 User 上传 $ns_{u_t}\|Token_{u_t,ns_i}\|C'_m$ 时,匹配 $ns_{u_t}\|Token_{u_t,ns_i}$ 和 C_t,若匹配成功,则意味着用户状态约

束条件是满足的,即可接受 User 在状态 ns_{u_t} 下提交的密文,将 C'_{m} 覆盖 C_{m}.

3.2 安全要求

1) 防止共谋. 2 个合法用户不能通过合并其属性私钥而提升原本单独不具备的读权限. 另外,假设用户 A 为了避免责任,不会将代表了写权限的凭证 $Token_{u_t,ns_i}$ 共享给用户 B,即每个用户为了自身利益考虑,不会将代表了责任的 $Token_{u_t,ns_i}$ 分享给他人.

2) 数据机密性. Owner 提交的数据在服务器端加密存储,Server 或非授权用户无法获知数据的内容.

3) 不可预测性. 假设 Server 是半可信的,即,Server 会忠实地完成代理匹配写权限的任务,但是对匹配的内容敏感. 同时,Server 不可预测每个用户是否可写,除非该 User 提交了正确的凭证.

3.3 安全模型的定义

定义 1. 如果在多项式时间内,敌手胜出下述安全游戏的最大优势是可忽略的,则所提出的方案是安全的.

初始化阶段. 挑战者运行 *Initial* 算法生成公共参数和主钥. 公共参数公开给敌手,主钥秘密保存在 AA 上.

阶段 1. 敌手通过向挑战者提交属性集 $I_{u_t,k}$ 查询属性私钥 $SK_{u_t,k}$. 挑战者运行 *Setup* 算法为敌手生成属性私钥 $SK_{u_t,k}$. 阶段 1 可以重复多次.

挑战阶段. 敌手提交 2 个消息:m_0 和 m_1,并给出读权限控制树 $policy_r$,但该控制树不能被阶段 1 中所提交的 $I_{u_t,k}$ 匹配. 接着,挑战者抛硬币选择 $b\in\{0,1\}$,并运行 *EncryptData* 算法输入 $policy_r$ 加密 m_b,得到加密结果 c_b,然后将 c_b 发送给敌手.

阶段 2. 过程与阶段 1 相同,只是添加了一条限制:$policy_r$ 不能被所挑战的属性集匹配. 敌手也可以询问 $SK_{u_t,k}$ 的解密委托密钥 $Part\_SK_{u_t}$,如此,敌手就可从挑战者获得 $Part\_SK_{u_t}$.

猜测阶段. 敌手输出猜想结果 $b'\in\{0,1\}$,如果 $b'=b$,则敌手胜出.

敌手胜出该游戏的优势 Adv 定义为

$$Adv=|Pr[b=b']-0.5|.$$

4 多用户协同编辑访问控制方法

4.1 基本思想

采用 CP-ABE 方法对外包数据进行访问控制时,Owner 不希望永远在线,并且本地存储容量有限,所以 Owner 在指定读权限访问策略 $policy_r$ 并加密数据后,立即将密文上传云服务器进行存储,以便其他用户可以随时随地读取数据或者编辑数据.

多个用户从云端下载密文后,用自己的私钥解密并读取数据,接着,用户也可以编辑数据,并将修改数据提交给云,云有 2 种处理方法:1)Owner 审核多个用户提交的数据编辑内容,并决定数据的最新版本. 基于这种处理方法,设计的多用户协同控制方法详见前期研究成果[18]. 2)云根据 Owner 定义的协同编辑控制策略决定数据的最新版本. 本文基于这种处理方法,设计多用户协同编辑访问控制方法,如图 2 所示. 为研究方便,暂不考虑并发机制的实现.

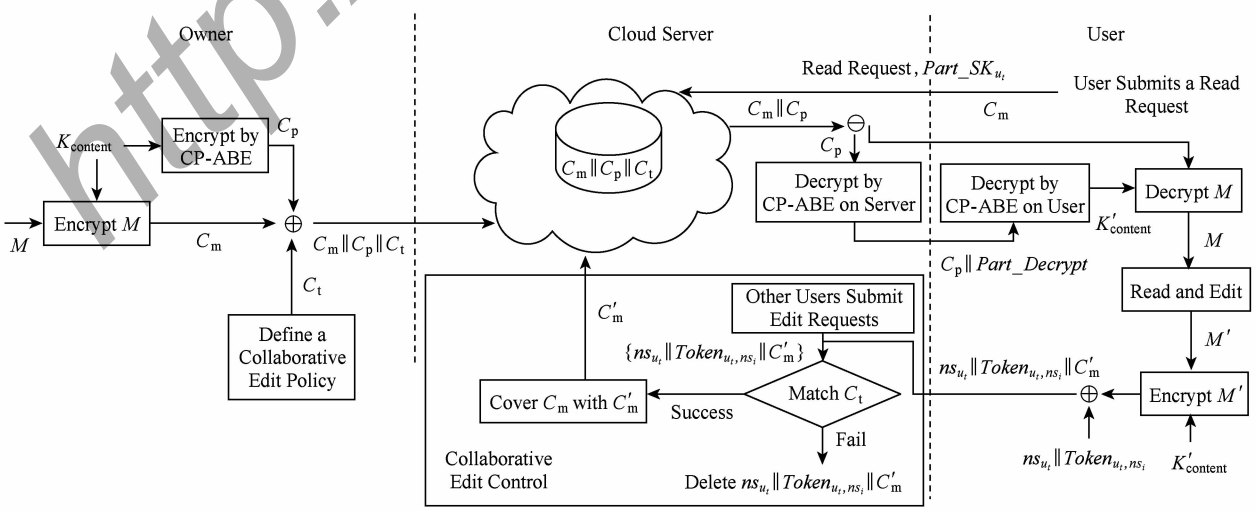


Fig. 2 The proposed framework for collaborative edit access control

图 2 多用户协同编辑访问控制框架

为了控制云端同一个密文被多个用户进行协同编辑,Owner 定义密文的写权限控制策略 C_t,并将 C_t 连同密文一起上传云端,由云辅助 Owner 进行写权限策略匹配. 密文数据的一致性由 Owner 定义的 C_t 来决定,即 Owner 可以通过任意定义 C_t 指定一致性规则. 这种将数据与策略绑定并且策略决定数据访问权限的方法称为 Data-binding-policy,适应于云存储等分布式系统的数据访问控制.

读权限控制要求云不能获知明文数据. 以往的 CP-ABE 方案中,用户下载密文后在本地解密,因而可以满足"云不能获知明文数据"这一安全要求. 为了减少用户端计算代价,将大部分解密工作委托给云服务器,并且仍然保证"云不能获知数据". 本文设计了用户委托解密密钥,能够让云使用该密钥完成大部分的解密工作,同时,该密钥又不能破坏数据的机密性.

与读权限控制不同,协同编辑权限控制要求:云不能获知数据内容,不能感知匹配内容,甚至也不能通过先前的匹配结果预测后续匹配的结果. 同时,读策略 $policy_r$ 采用的布尔公式描述方法也不适应于协同编辑策略 C_t,这是因为布尔公式比较适合描述属性是否配对,无法描述复杂的控制需求(见本文引言部分).

复杂的协同编辑控制策略涉及到数值大小的比较(例如时间的先后比较等),需要更丰富的表达能力(例如多用户之间的状态条件). 相对于布尔公式,circuit 具有复杂控制需求的描述能力,因而本文采用 circuit 描述协同编辑控制策略.

关于协同编辑控制策略的云辅助匹配,本文采用 LWE 理论的矩阵运算,而不是 $policy_r$ 匹配时的双线性映射运算. 这是因为采用类似于 $policy_r$ 匹配时的双线性映射运算进行匹配时,需要将协同编辑策略映射为访问控制树,将用户权限映射为属性集,然后在云服务器上进行双线性映射匹配. 这种方法存在 2 个问题:1)云在匹配运算之前就能预测下一个用户提交数据是否会被接受;2)匹配效率低. 假设用户数量为 N,用户状态的数量为 M. 极端情况下,需要对每个用户在每个状态下访问数据时指定一个控制树,每个树的属性数目是 $N\times M$,则每个数据的协同编辑策略需要 $N^2\times M^2$ 个属性来定义. 假如用户用数量为 K 的属性集来表示,则需要 $K^2\times M^2$ 个属性来定义协同编辑策略.

考虑上述安全及效率问题,本文设计了基于 LWE 假设的 MOR 原语和新的协同编辑控制策略匹配方法,利用 MOR 原语的不可预测性和可重用性,使得本文提出的方法既能让云高效地完成写权限的辅助匹配,又能保证云在匹配运算之前无法预测用户提交的修改数据是否被接受(即云的不可预测性).

4.2 详细工作流程

各个阶段运行不同算法,如图 3 所示.

1) 初始化阶段(initialization phase). AA 调用 *Initial* 算法,生成全局公共参数 *params* 和主钥 $MK=(MK_1,MK_2)$,将 *params* 和 MK_2 发送给申请注册的 Owner. 接着,AA 调用 *Setup* 算法,为用户生成属性私钥 SK_{u_t}、云辅助解密密钥对(DPK_{u_t}, DSK_{u_t}),将 SK_{u_t},DPK_{u_t},DSK_{u_t} 通过密钥交换协议发送给 User. 然后,AA 调用 *SetupCollaborativeEdit* 算法生成协同编辑控制参数 $\boldsymbol{mpk}_i$,$\boldsymbol{msk}_i$,$\boldsymbol{pk}_{\text{out}}$,并将 $\boldsymbol{mpk}_i$ 和 $\boldsymbol{pk}_{\text{out}}$ 广播给 Owner,将 $\boldsymbol{msk}_i$ 通过密钥交换协议发送给 User. 值得注意的是,属性字符串在 *Setup* 算法中由 AA 任意指定,无需在 *Initial* 算法以枚举类型罗列出来,即属性域无需明确定义,这样设计灵活性更强.

2) 数据准备阶段(data preparing phase). Owner 调用 *EncryptData* 算法,用对称密钥 K_{content} 对明文数据 M 运行对称密码算法进行加密得到 C_{m},Owner 定义具有读权限的策略 $policy_r$,用 $policy_r$ 对对称密钥 K_{content} 进行 CP-ABE 加密得到 C_{p},将 $C_{\text{m}}\|C_{\text{p}}$ 上传到服务器.

定义协同编辑条件(collaborative edit policy definition),Owner 运行 *EncryptCollaborativeEdit* 算法,定义协同编辑条件 C_t,上传到服务器,服务器将 C_t 附在 $C_{\text{m}}\|C_{\text{p}}$ 后面,得到 $C_{\text{m}}\|C_{\text{p}}\|C_t$ 后存储.

3) 数据读取阶段(data read phase). 当 User 向服务器申请数据时,User 提交其 $Part\_SK_{u_t}$,服务器使用用户提交的 $Part\_SK_{u_t}$,运行 *DecryptNode* 算法计算出 *Part_Decrypt*,然后将 *Part_Decrypt* 连同 $C_{\text{m}}\|C_{\text{p}}$ 一起发送给用户,用户运行 *DecryptData* 算法使用 SK_{u_t} 和 C_{p} 获得 K_{content},并使用 K_{content} 和 C_{m} 运行对称解密算法,即可读取明文.

4) 数据编辑阶段(data edit phase). 当 User 修改数据之后,提交 $ns_{u_t}\|Token_{u_t,ns_i}\|C'_{\text{m}}$ 到服务器,服务器运行 *MatchCollaborativeEdit* 算法匹配 C_t,若匹配成功,则接受修改,用 C'_{m} 覆盖 C_{m}.

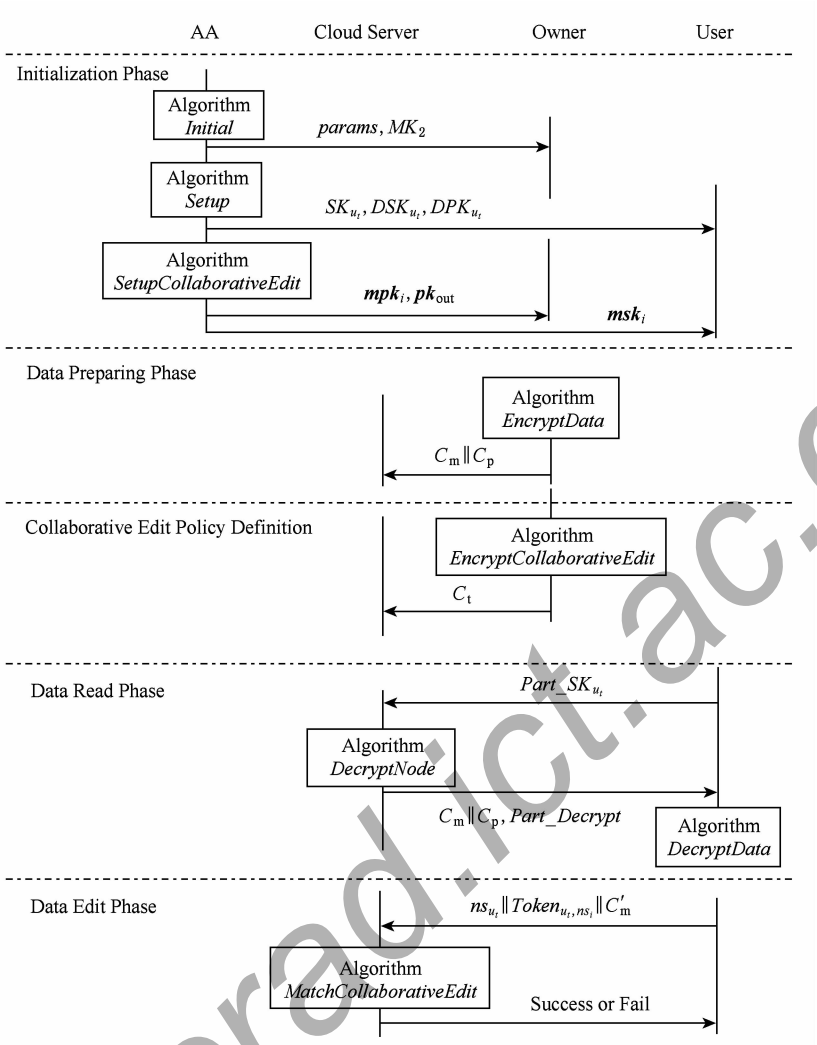


Fig. 3 Phases and algorithms

图 3 阶段与算法

4.3 核心算法

G 和 G_T 表示具有素数阶 p 的双线性群. g 表示群 G 的生成元. $e:G\times G\rightarrow G_T$ 表示双线性映射. $H:\{0,1\}^*\rightarrow G$ 表示二进制数到群元素映射的 Hash 函数,$\widetilde{H}:G\rightarrow\{0,1\}^n$ 表示群元素到二进制数映射的 Hash 函数.

1) *Initial* 算法

Initial 算法运行在 AA 上. AA 选择双线性群 G 和 G_T,g 为 G 的生成元,选择 $e:G\times G\rightarrow G_T$ 为双线性映射,选择 Hash 函数 $H:\{0,1\}^*\rightarrow G$,$\widetilde{H}:G\rightarrow\{0,1\}^n$. AA 生成公共参数 *param* 和主钥 MK:

$$params=(g,G,G_T,e,H,\widetilde{H},PP),$$

$$MK=(MK_1=g^a,MK_2=e(g,g)^b),$$

其中 $a,b\in\mathbb{Z}_p^*$.

2) *Setup* 算法

Setup 算法运行在 AA 上. AA 接受用户的注册,并为每个合法的用户生成密钥. 假设每个用户 u_t 的属性集为 I_{u_t},AA 对用户 u_t 的每个属性选择随机数 $r_1,r_2,\cdots,r_j,\cdots\in\mathbb{Z}_p^*$,生成用户私钥 SK_{u_t}:

$$SK_{u_t}=(D,\{(D_j,D_j')\}_{\lambda_j\in I_{u_t}}),$$

$$D=MK_1\ g^{r_t}=g^a\ g^{r_t},$$

$$D_j=g^{r_t}\ H(\lambda_j)^{r_j},$$

$$D_j'=g^{r_j}.$$

为了完成云辅助解密,AA 还要为 User 生成委托解密的密钥对:

选择随机数 $r_u\in\mathbb{Z}_p^*$,计算 $DPK_{u_t}=g^{r_u+b}$,$DSK_{u_t}=g^{r_u+a}$.

AA 通过密钥交换协议将 SK_{u_t},DSK_{u_t},DPK_{u_t} 发送给用户.

3) *SetupCollaborativeEdit* 算法

AA 为每一个参与协同编辑的用户生成 2 个密钥向量:

$$\boldsymbol{mpk}_i=(\boldsymbol{pk}_{i,1},\boldsymbol{pk}_{i,2},\cdots,\boldsymbol{pk}_{i,M}),$$

$$\boldsymbol{msk}_i=(\boldsymbol{statusk}_{i,1},\boldsymbol{statusk}_{i,2},\cdots,\boldsymbol{statusk}_{i,M}),$$

$$\boldsymbol{pk}_{i,j}=\boldsymbol{A}_{i,j}\in\mathbb{Z}_q^{n\times m},$$

$$\boldsymbol{statusk}_{i,j}=\boldsymbol{T}_{i,j}\in\mathbb{Z}_q^{m\times m},$$

其中,$\boldsymbol{pk}_{i,j}$ 和 $\boldsymbol{statusk}_{i,j}$ 是通过运行后门生成算法 $TrapGen(1^n,1^m,q)$ 生成的矩阵 $\boldsymbol{A}_{i,j}$ 和后门矩阵 $\boldsymbol{T}_{i,j}$.

最后,AA 为协同编辑的用户集合生成一个状态控制公钥:$\boldsymbol{pk}_{\text{out}}=\boldsymbol{A}_{\text{out}}\in\mathbb{Z}_q^{n\times m}$.

AA 将 $\boldsymbol{mpk}_i$ 和 $\boldsymbol{pk}_{\text{out}}$ 广播于 Owner 用户,并将 $\boldsymbol{msk}_i$ 通过密钥交换协议分发给用户.

4) *EncryptData* 算法

EncryptData 算法运行在 Owner 上. 先用对称加密算法将明文 M 加密成 C_{m},然后加密密钥 K_{content} 用 CP-ABE 方法加密得到 C_{p},CP-ABE 加密时,使用数据读策略 $policy_r$. 接着 Owner 将 $C_{\text{m}}\|C_{\text{p}}$ 上传到服务器.

Owner 对数据进行读权限控制的 CP-ABE 加密过程如下:选择随机数 $s_{\text{c}}\in\mathbb{Z}_p^*$,令 $q_R(0)=s_{\text{c}}$,其中,R 表示 $policy_r$ 的根节点,$q_R(x)$ 表示根节点 R 的多项式. Owner 以嵌套的方式设置每个节点 x 的多项式 q_x. 每个节点 x 的门限为 k_x,q_x 的度设为 $d_x=k_x-1$. q_x 的常数项 $q_x(0)$ 设置为 $q_{parent(x)}(index(x))$,q_x 的其他项系数是随机选取的整数. C_{p} 的构造如下:

$$C_{\text{p}}=(policy_r,\hat{C},C_r,\{(C_i,C_i')\}_{\lambda_j\in S_{\text{T}}}),$$

$$\hat{C}=K_{\text{content}}\ MK_2=K_{\text{content}}\ e(g,g)^{b\cdot s_{\text{c}}},$$

$$C_r=g^{s_{\text{c}}},$$

$$C_i=g^{q_i(0)},$$

$$C_i'=H(\lambda_i)^{q_i(0)},$$

其中,S_{T} 是访问策略树 $policy_r$ 所有属性(即,叶子节点)的集合.

5) *EncryptCollaborativeEdit* 算法

EncryptCollaborativeEdit 算法运行在 Owner 上. Owner 将每个用户作为 circuit 的一个 input wire,并对于用户状态约束条件抽象化的合取范式抽象为 circuit,将 circuit 编码为 C_{t}:

$$C_{\text{t}}=(\boldsymbol{\varphi}_1,\boldsymbol{\varphi}_2,\cdots,\boldsymbol{\varphi}_N,\Gamma),$$

$$\boldsymbol{\varphi}_i=Encode(\boldsymbol{pk}_{i,ns_i},w),$$

$$\Gamma=E(Encode(\boldsymbol{pk}_{\text{out}},w),\tilde{H}(\hat{C})).$$

Owner 将 C_{t} 上传,并附于 $C_{\text{m}}\|C_{\text{p}}$ 之后,存储于云.

6) *MatchCollaborativeEdit* 算法

MatchCollaborativeEdit 算法运行在 Cloud Server 上. 令 $Token_{u_t,ns_i}=\boldsymbol{statusk}_{u_t,ns_i}$,当 User 提交 $ns_{u_t}\|Token_{u_t,ns_i}\|C_{\text{m}}'$ 时,云进行 C_{t} 匹配. 具体过程如下:

$$rk=RekeyGen(\{\boldsymbol{pk}_{i,ns_i}\},Token_{u_t,ns_i},\boldsymbol{pk}_{\text{out}}),$$

$$\boldsymbol{\varphi}_{\text{out}}=Recode(rk,\boldsymbol{\varphi}_1,\boldsymbol{\varphi}_2,\cdots,\boldsymbol{\varphi}_N),$$

若 $D(\boldsymbol{\varphi}_{\text{out}},\Gamma)=\tilde{H}(\hat{C})$,则匹配成功,Server 接受用户提交的 C_{m}',用 C_m' 覆盖 C_{m}.

为了确保 Cloud 对编辑请求的不可预测性,需要在每一次成功覆盖时重新刷新 circuit.

重新刷新 circuit 的过程如下:选择随机数 s',令 $\boldsymbol{\varphi}_i=\boldsymbol{\varphi}_i s'$,$\Gamma=\Gamma s'$. 第 i 次刷新 circuit 时,选择随机数 $t_i\in\mathbb{Z}_p^*$,令 s_i 代表第 i 次选取的 s',$s_i=(1+(-1)^{(t_i+1)}(1/i))/i$,详见本文 5.1 节.

7) *DecryptData* 算法

DecryptData 算法运行在 User 上,使用用户的私钥 SK_{u_t} 解密密文中的 C_{p},如果用户的属性集 I_{u_t} 满足 C_{p} 中关联的访问策略,则解密成功,获得对称密钥 K_{content},继而通过对称解密算法解密 C_{m},获得明文 M.

使用 SK_{u_t} 解密 C_{p} 的过程如下:

用户将 DSK_{u_t} 附到 D_j 后,得到 $Part\_SK_{u_t}$:

$$Part\_SK_{u_t}=\{(\tilde{D}_j,D_j')\}_{\lambda_j\in I_{u_t}},$$

$$\tilde{D}_j=D_j\ DSK_{u_t}=g^{r_t}\ H(\lambda_j)^{r_j}\ g^{r_u+a},$$

$$D_j'=g^{r_j}.$$

用户将 $Part\_SK_{u_t}$ 发送给服务器,请求其辅助解密,服务器收到用户请求后,从访问策略树 $policy_r$ 的根节点开始,以递归方式调用 *DecryptNode* 算法. 假设用户私钥中的属性集为 I_{u_t},$policy_r$ 的叶子节点集合为 I_{Tree},λ_i 为节点 x_i 的属性. *DecryptNode* 算法描述如下:

算法 1. *DecryptNode*.

输入:C_{p},$Part\_SK_{u_t}$,I_{u_t};

输出:$Part\_Decrypt$;

If $x_i\in I_{\text{Tree}}$ Then

If $\lambda_i\in I_{u_t}$ Then

$$Part\_Decrypt=\frac{e(\tilde{D}_i,C_i)}{e(D_i',C_i')}=\frac{e(g^{r_t}\ g^{r_u+a}\ H(\lambda_i)^{r_i},g^{q_i(0)})}{e(g^{r_i},H(\lambda_i)^{q_i(0)})}=e(g,g)^{(r_t+r_u+a)\,q_i(0)};$$

End If

If $\lambda_i\notin I_{u_t}$ Then

$Part\_Decrypt=\perp$;

End If

End If

If $x_i\in policy_r$ and $x_i\notin I_{\text{Tree}}$ Then

$$Part\_Decrypt_x=\prod_{z\in I_x}Part\_Decrypt_z^{\Delta_{i,s_x}(0)}=e(g,g)^{(r_t+r_u+a)\cdot q_x(0)};$$

End If

If $x_i==root(policy_r)$ Then

Return $Part\_Decrypt$;

End If

其中,$Part\_Decrypt_z$ 表示节点 x_i 的孩子 z 上 $DecryptNode$ 算法的运算结果. $i=index(z)$, $s_x=\{index(z),z\in I_{x_i}\}$, I_{x_i} 表示节点 x_i 的所有孩子节点组成的集合. 如果 $policy_r$ 匹配成功,即可获得: $Part\_Decrypt=e(g,g)^{(r_t+r_u+a)\cdot s_c}$.

服务器将 $Part\_Decrypt$ 连同 $C_{\text{m}}\|C_{\text{p}}$ 一起发送给用户 u_t. 用户从服务器获得 $Part\_Decrypt$ 和 $C_{\text{m}}\|C_{\text{p}}$ 后,计算 K_{content}:

$$\hat{D}=D\times DPK_{u_t}=g^a\,g^{r_t}\,g^{r_u+b},$$

$$\frac{\hat{C}\times Part\_Decrypt}{e(C,\hat{D})}=K_{\text{content}}.$$

用户得到内容密钥 K_{content} 后,解密 C_{m},读取明文 M.

5 分 析

5.1 正确性

1) MOR 原语的正确性分析

当 $\boldsymbol{e}$ 服从 Error 分布(即 $\boldsymbol{e}\sim X^m$)时,根据界 B 的定义,得知 $\|\boldsymbol{e}\|_\infty\leqslant B$. 又因为 $\mathbb{Z}^m$ 上的截断离散高斯分布(truncated discrete Gaussian distribution) $D_{\boldsymbol{\Gamma}^m,s}$ 要求 $\|\cdot\|_\infty\leqslant w\sqrt{m}$,因而随机算法 $SampleD$ 输出向量的无穷范式必须小于等于 $w\sqrt{m}$. 又因为 $Recode$ 算法的 $\boldsymbol{\varphi}_{\text{tgt}}=[\boldsymbol{\varphi}_1^{\text{T}}\|\boldsymbol{\varphi}_2^{\text{T}}\|\cdots\|\boldsymbol{\varphi}_N^{\text{T}}]\,rk^{\text{tgt}}$ 计算过程会使 $\boldsymbol{e}$ 部分增加,因而第 j 次运行 $Recode$ 算法时,要求 $\|\boldsymbol{e}\|_\infty\leqslant B(2mw\sqrt{m})^j$.

$Recode$ 算法在计算 $\boldsymbol{\varphi}_{\text{tgt}}$ 时,要保证 $\boldsymbol{e}$ 值足够小,即满足 $\|\boldsymbol{e}\|_\infty\leqslant B(2mw\sqrt{m})^j$. 分析如下:令 $\boldsymbol{\varphi}_1$, $\boldsymbol{\varphi}_2$, …, $\boldsymbol{\varphi}_N$ 为 N 个输入编码,令 $\boldsymbol{\varphi}_{\text{tgt}}$ 为输出编码,则 $\boldsymbol{\varphi}_{\text{tgt}}^{\text{T}}=([\boldsymbol{\varphi}_1^{\text{T}}\|\boldsymbol{\varphi}_2^{\text{T}}\|\cdots\|\boldsymbol{\varphi}_N^{\text{T}}]\,rk^{\text{tgt}})^{\text{T}}=\boldsymbol{s}^{\text{T}}\boldsymbol{A}_{\text{tgt}}+\boldsymbol{e}_{\text{tgt}}$,其中, $\boldsymbol{e}_{\text{tgt}}=[\boldsymbol{e}_1^{\text{T}}\|\boldsymbol{e}_2^{\text{T}}\|\cdots\|\boldsymbol{e}_N^{\text{T}}]\,rk^{\text{tgt}}$,则 $\|\boldsymbol{e}_{\text{tgt}}\|_\infty\leqslant m\,\|rk^{\text{tgt}}\|_\infty\times(\|\boldsymbol{e}_1\|_\infty+\|\boldsymbol{e}_2\|_\infty+\cdots+\|\boldsymbol{e}_N\|_\infty)$,又因为 rk^{tgt} 作为 $SampleD$ 的输出,因而 $\|rk^{\text{tgt}}\|_\infty\leqslant w\sqrt{m}$,因而有: $\|\boldsymbol{e}_{\text{tgt}}\|_\infty\leqslant mw\sqrt{m}\,(\|\boldsymbol{e}_1\|_\infty+\|\boldsymbol{e}_2\|_\infty+\cdots+\|\boldsymbol{e}_N\|_\infty)$. 假设 $\boldsymbol{\varphi}_1$ 的编码级别是 j_1, $\boldsymbol{\varphi}_2$ 的编码级别是 j_2, $\boldsymbol{\varphi}_N$ 的编码级别是 j_N, $\boldsymbol{\varphi}_{\text{tgt}}$ 的编码级别为 j,则有 $\|\boldsymbol{e}_1\|_\infty\leqslant B(2mw\sqrt{m})^{j_1}$ 和 $\|\boldsymbol{e}_2\|_\infty\leqslant B(2mw\sqrt{m})^{j_2}$, …, $\|\boldsymbol{e}_N\|_\infty\leqslant B(2mw\sqrt{m})^{j_N}$,则 $\|\boldsymbol{e}_{\text{tgt}}\|_\infty\leqslant mw\sqrt{m}\,(B(2mw\sqrt{m})^{j_1}+\cdots+B(2mw\sqrt{m})^{j_N})$,即 $\|\boldsymbol{e}_{\text{tgt}}\|_\infty\leqslant B(2mw\sqrt{m})^{\max(j_1,j_2,\cdots,j_N)+1}$. 又因为 $j_1,j_2,\cdots,j_N<j$,所以 $\max(j_1,j_2,\cdots,j_N)+1\leqslant j$ 成立,因而 $\|\boldsymbol{e}_{\text{tgt}}\|_\infty\leqslant B(2mw\sqrt{m})^j$,符合要求.

另一方面,为了保证对称加密算法 (E,D) 的正确性[4],2 个编码 $\boldsymbol{\varphi}_0=\boldsymbol{A}^{\text{T}}\boldsymbol{s}+\boldsymbol{e}_0$ 和 $\boldsymbol{\varphi}_1=\boldsymbol{A}^{\text{T}}\boldsymbol{s}+\boldsymbol{e}_1$,必须满足 $\|\boldsymbol{\varphi}_0-\boldsymbol{\varphi}_1\|_\infty<q/4$,即 $\|\boldsymbol{\varphi}_0-\boldsymbol{\varphi}_1\|_\infty=\|\boldsymbol{e}_0-\boldsymbol{e}_1\|_\infty\leqslant 2B(2mw\sqrt{m})^{d_{\max}}<q/4$.

分析完毕.

2) 多用户协同编辑控制方法的正确性分析

本文提出的多用户协同编辑控制方法是基于 MOR 原语设计的,因而其核心算法是正确的,但存在退化性: $EncryptCollaborativeEdit$ 算法中涉及到的 Error 参数均需满足 $\|\boldsymbol{e}\|_\infty\leqslant B(2wm\sqrt{m})^j$,则 $MatchCollaborativeEdit$ 算法运行时,刷新 circuit 时 Error 部分也必须要满足 $\|\boldsymbol{e}\|_\infty\leqslant B(2wm\sqrt{m})^j$.

下面分别阐述刷新 circuit 对 $\boldsymbol{\varphi}_i$ 和 Γ 带来的影响,为方便表达,2 个向量相乘的结果就是对应位上元素相乘得到的向量,向量与数字相乘就是向量的每一个元素都乘以该数字.

$\boldsymbol{\varphi}_i\leftarrow\boldsymbol{\varphi}_i\,s'$ 时, $\boldsymbol{\varphi}_i=\boldsymbol{A}_i^{\text{T}}\boldsymbol{s}\,s'+\boldsymbol{e}\,s'$,由于 s' 为随机数,因而必须得满足 $\boldsymbol{e}\,s'\approx\boldsymbol{e}\leqslant B(2wm\sqrt{m})^j$.

$\Gamma\leftarrow\Gamma\,s'$ 时,

$$\Gamma=E(Encode(\boldsymbol{pk}_{\text{out}},w),\tilde{H}(\hat{C}))\,s'=Encode(\boldsymbol{pk}_{\text{out}},w)\,s'+\lceil q/2\rfloor\tilde{H}(\hat{C})\,s'(\text{mod}\,q)=\boldsymbol{pk}_{\text{out}}^{\text{T}}\boldsymbol{s}\,s'+\boldsymbol{e}\,s'+\lceil q/2\rfloor\tilde{H}(\hat{C})\,s'(\text{mod}\,q),$$

必须得满足 $\lceil q/2\rfloor\tilde{H}(\hat{C})\,s'\approx\lceil q/2\rfloor\tilde{H}(\hat{C})$.

如上所述,为了满足正确性和可重用性,避免退化性,令 s_i 代表第 i 次选取的 s',要求 s_i 在多次选取时满足: 1) s_i 随机选取; 2) 多次选取的 s_i 连乘后约等于 1. 为了仿真 s' 选取方法,令 $Y_i=\prod_{i=[1,30]}s_i$. 仿真 Y_i 如图 4 所示:

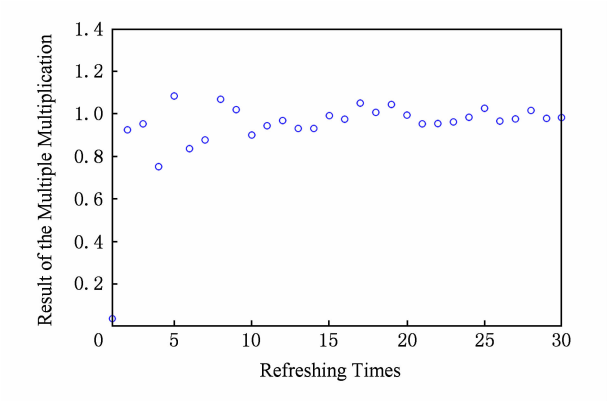


Fig. 4 Reusable parameters

图 4 可重用参数

5.2 安全性分析与证明

1) 抗共谋分析

即使半可信云与恶意用户之间共谋,也不能让恶意用户获得更多权限. 云不能通过合并用户 A 和用户 B 的 $Part\_SK_{u_t}$,以便让用户 A 获得更多权限.

假设用户 A 的属性私钥为 $SK_{u_t}^{A}$,用户 B 的属性私钥为 $SK_{u_t}^{B}$,合并用户 B 的私钥到用户 A 上之后,用户 A 的私钥为

$$SK_{u_t}^{AB}=(D,\{(D_j,D_j')\}_{\lambda_j\in I_{u_t}^{A}}\cup\{(D_j,D_j')\}_{\lambda_j\in I_{u_t}^{B}}\}=$$
$$(g^{a}g^{r_t^{A}},\{(g^{r_t^{A}}H(\lambda_j)^{r_j},g^{r_j^{A}})\}_{\lambda_j\in I_{u_t}^{A}}\cup$$
$$\{(g^{r_t^{B}}H(\lambda_j)^{r_j},r_t^{B})\}\},$$

其中,r_t^{A} 和 r_t^{B} 是 AA 为用户 A 和用户 B 分别选取的随机数. 为了解密 C_{p},$Part\_Decrypt$ 在 $SK_{u_t}^{A}$ 上运行得到 $e(g,g)^{(r_t^{A}+r_u^{A}+a)s_{\mathrm{c},A}}$,在 $SK_{u_t}^{B}$ 上运行得到:$Part\_Decrypt=e(g,g)^{(r_t^{B}+r_u^{B}+a)s_{\mathrm{c},B}}$,由于加密时 $s_{\mathrm{c},A}\neq s_{\mathrm{c},B}$,所以无法通过合并用户私钥得到更多权限.

2) 数据机密性安全分析

明文通过对称加密算法加密成 C_{m} 存储于服务器,而对称加密时的密钥 K_{content} 也用 $policy_r$ 加密成 C_{p} 放在服务器. 只有密钥携带的属性集与 $policy_r$ 匹配的用户才可以解密 C_{p},得到 K_{content},从而使用 K_{content} 从 C_{m} 中恢复出明文. 非授权用户和服务器即使获得 C_{m} 和 C_{p},也不能够恢复明文 M.

3) 不可预测性分析

Server 不能通过多次输入 $(L_1,L_2,\cdots,L_N)$ 记录 $Encode(G(L_1,L_2,\cdots,L_N),w)$ 提升共谋用户的编辑权限. 分析如下:MOR 原语具有不可预测性,也就是说,$Encode(G(L_1,L_2,\cdots,L_N),w)$ 的计算结果不增加计算 $Encode(1-G(L_1,L_2,\cdots,L_N),w)$ 的优势,即,当且仅当 $G(L_1,L_2,\cdots,L_N)=1$ 成立时,才能获得正确的 $Encode(G(L_1,L_2,\cdots,L_N),w)$. 多次记录 $G(L_1,L_2,\cdots,L_N)=0$ 时的 $Encode(G(L_1,L_2,\cdots,L_N),w)$ 结果是无用的,因而云对每一次提交的写请求结果具有不可预测性;又因为 MOR 原语具有可重用性,因而可以在每一次成功写之后实现转换表的刷新. 综合上述 2 条原因,先验知识不能帮助敌手获得更多的优势.

4) 数据机密性安全证明

在 Bethencourt 等人[19]的基础上,本文构造了读权限访问控制方法:云辅助解密时,用户需要将 $Part\_SK_{u_t}$ 发送给服务器. 接下来,证明在离散对数困难问题假定下,$Part\_SK_{u_t}$ 暴露给敌手并不能增加其胜出安全游戏的优势.

首先,证明敌手不能从 $\tilde{D}_j$ 和 D_j' 获得 D:根据离散对数困难问题假定,敌手不能从 $D_j'=g^{r_j}$ 中计算 r_j,因而根据 g^{r_j} 无法计算 $H(\lambda_j)^{r_j}$,继而敌手不能从 $params$ 和 $\tilde{D}_j$ 中推导出 $H(\lambda_j)^{r_j}g^{r_u}$,所以敌手不能从 $\tilde{D}_j=g^{r_t}H(\lambda_j)^{r_j}g^{r_u+a}$ 和 $D_j'=g^{r_j}$ 推导出 $D=g^{a}g^{r_t}$.

接下来,证明敌手不能从 $\tilde{D}_j$ 和 D_j' 获得 D_j:根据设计,$PSK_{u_t}=g^{r_u+a}$ 是从 AA 通过密钥交换协议分发给 User 的,不暴露给敌手,因而 g^{r_u+a} 对于敌手是保密的,因而敌手无法从 $\tilde{D}_j=g^{r_t}H(\lambda_j)^{r_j}g^{r_u+a}$ 推导出 $D_j=g^{r_t}H(\lambda_j)^{r_j}$.

结论:即使 Adversary 获得 $Part\_SK_{u_t}$,也不能从 $\tilde{D}_j$ 和 D_j' 获得 D 和 D_j.

5) 用户编辑权限控制分析

授权用户可以得到写权限:根据构造后门应具有的安全特性(文献[20]中命题 5.1),矩阵及其后门是一一对应的,拥有后门的用户可以获得正确的解密结果. 本文方法中,$SetupCollaborativeEdit$ 算法运行后,每一个参与协同编辑的用户从 AA 获得 $\boldsymbol{msk}_i$,这个向量的每一和个分量对应了后门算法生成的后门矩阵 $\boldsymbol{T}_{i,j}$. 另外,Owner 运行 $EncryptCollaborativeEdit$ 算法生成的 C_{t} 是以 $\boldsymbol{mpk}_i$ 和 $\boldsymbol{pk}_{\mathrm{out}}$ 为参数构造的,$\boldsymbol{mpk}_i$ 的每一个分量对应了后门算法的矩阵 $\boldsymbol{A}_{i,j}$,根据本文 5.1 节 MOR 原语和本文方法的正确性分析,拥有写权限的用户具有合适的凭证 $Token_{u_t,ns_i}$,会得到成功的 C_{t} 匹配结果.

非授权用户不可得到写权限. 用户提交的 $Token_{u_t,ns_i}$ 代表了写权限的凭证,是与 $\boldsymbol{pk}_{i,j}$ 对应的后门矩阵. 按照后门构造具体参数要求(文献[20]中命题 5.3)和 $SampleD$ 函数抽样要求(文献[20]中引

理 4.2),能够在格困难问题 GapSVP$_\gamma$,SIS,SIVP$_\gamma$ 的假设下保证非授权用户得不到正确的写权限.同时,在 SIS$_{q,m,2w\sqrt{m}}$ 困难问题假设下满足后门函数防碰撞要求.

5.3 读权限控制的代价分析

1) 存储代价分析

表 1 给出了已有方法[2,21]与本文方法之间存储开销方面的比较.

表 1 中,L_G 表示群 G 中一个元素的存储量;$n_{a,k}$ 表示授权机构 AA_k 管理下的属性数量;n_{a,k,u_t} 表示授权机构 AA_k 为用户 u_t 生成的私钥中关联的属性数量;$n_{u_t,k}$ 表示授权机构 AA_k 管理的用户数量;t_r 表示 Owner 指定的 $policy_r$ 中的属性数量;N_A 表示密文中关联的 AA 数量;L_S 表示密文 C_t 中的存储量.与文献[2,21]已有方法相同,表 1 忽略随机整数的存储代价.

Table 1 Comparison of Storage Cost

表 1 存储代价比较

Methods	Owner	User	Cloud Server
Ref[2]	$(2N_A+\sum_{k=1}^{N_A} n_{a,k})L_G$	$(3N_A+1+\sum_{k=1}^{N_A} n_{a,k,u_t})L_G$	$(3+3t_r)L_G$
Ref[21]	$(3N_A+1+\sum_{k=1}^{N_A} n_{a,k})L_G$	$(1+2\sum_{k=1}^{N_A} n_{a,k,u_t})L_G$	$(3+3t_r)L_G$
Our Method	$(1+\sum_{k=1}^{N_A} n_{a,k})L_G$	$(2+2\sum_{k=1}^{N_A} n_{a,k,u_t})L_G$	$(2+2t_r)L_G+L_S$

由表 1 可知,本文方法在 Owner 上的存储代价都比较小,适合轻量级终端应用.但本文方法在服务器上的存储代价与密文中状态约束控制所用的 C_t 相关,这是因为本文方法考虑了多用户协同编辑控制,而其他 2 个方法只考虑了读权限控制.

2) 计算代价分析

表 2 比较了已有方法[2,21]与本文方法之间核心算法运行时的计算代价.表 2 中,E_G 表示群 G 上指数运算的计算量;P_G 表示群 G 上双线性映射运算的计算量;k 表示用户属性私钥携带的属性数量;I_{A_k} 表示密文中关联的授权机构 AA_k 管理的属性数量.与文献[2,21]已有方法相同,表 2 忽略乘法与除法的计算代价.

Table 2 Comparison of Computation Cost

表 2 计算代价比较

Methods	Algorithm Encrypt (on Owner)	Algorithm Decrypt (on User)	Algorithm Decrypt(on Cloud/Server)
Ref[2]	$(2+2t_r)E_G$	$(k+\log t_r)E_G+(2k+1)P_G$	0
Ref[21]	$(3+6t_r)E_G$	E_G	$N_A\times((\sum_{k=1}^{I_{A_k}}(3P_G+E_G))+2P_G)$
Our Method	$(2+2t_r)E_G$	E_G	$(k+\log t_r)E_G+2kP_G$

由表 2 可知,本文方法的用户端(包括 Owner 和 User)在加密解密时的计算量都比较小,适合轻量级终端应用.同时,文献[2,21]只考虑了单用户独立访问读取数据的情景.

6 仿 真

6.1 读权限访问控制方法的仿真实验

仿真实验运行平台为 Ubuntu 操作系统,双线性映射运算函数库采用 PBC(pairing based cryptography library),大整数数据库采用 GMP (GNU MP Bignum library).数据加密和解密过程中用户端的计算代价仿真如图 5 所示,椭圆曲线选择 α,群的阶均为 160 b,域的大小为 512 b.$policy_r$ 中的属性个数和用户属性私钥中所含属性的个数默认设置为 20.为了使实验结果稳定,运行时间取 20 次运行的平均.

图 5(a)是 Owner 端加密过程仿真结果,可以看到,3 个方法的加密时间与 $policy_r$ 中的属性个数都成线性关系,同时,文献[21]的加密代价明显高于本文所提方法和文献[2],这是因为文献[21]中的加密是计算 $C,C',C'',\{C_i,D_{1,i},D_{2,i}\}$,计算 C,C',C''分别需要 1 次指数运算.计算 $C_i,D_{1,i},D_{2,i}$ 分别需要 2 次、1 次和 1 次指数运算.

图 5(b)(c)是 User 端解密过程仿真结果,可以看到,$policy_r$ 中的属性个数变化对 User 端解密时间影响不大.文献[2]的 User 端解密时间明显高于

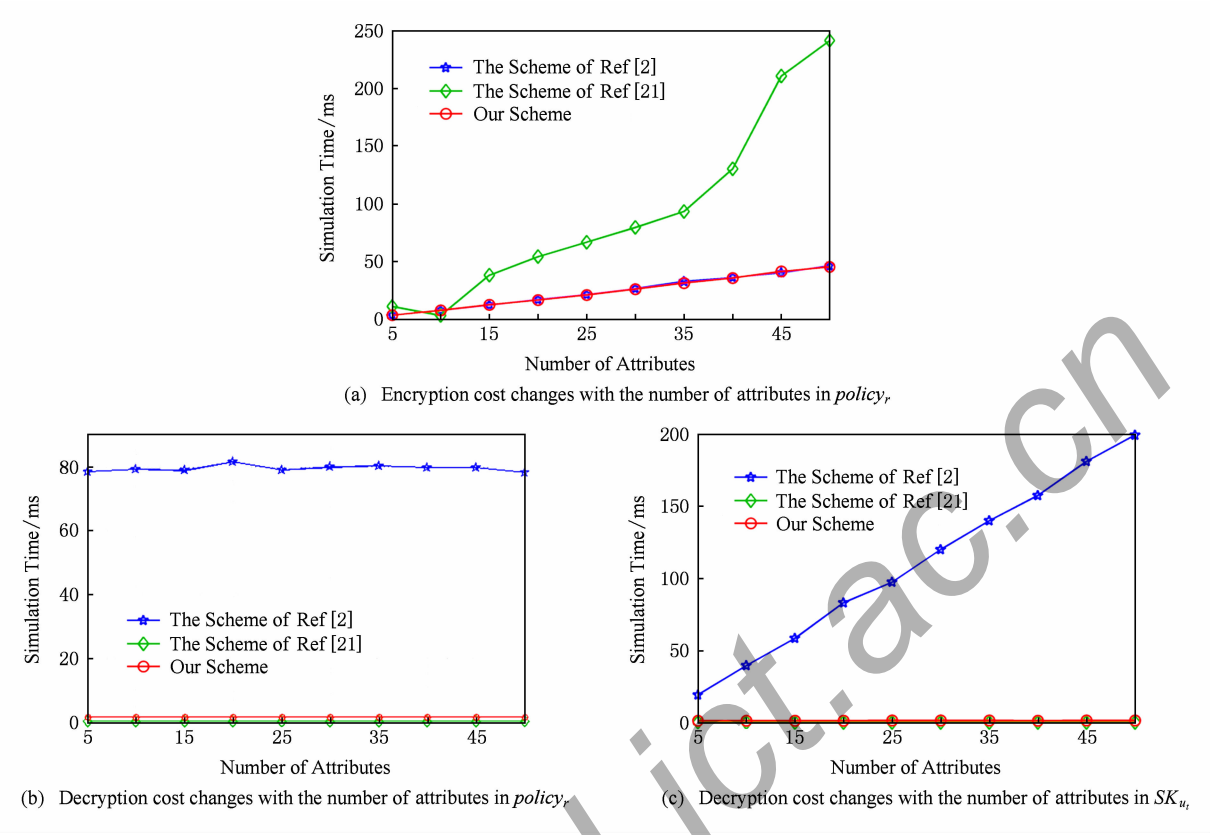


Fig. 5 Simulation of encryption and decryption costs on User

图 5 用户端加解密计算代价的仿真

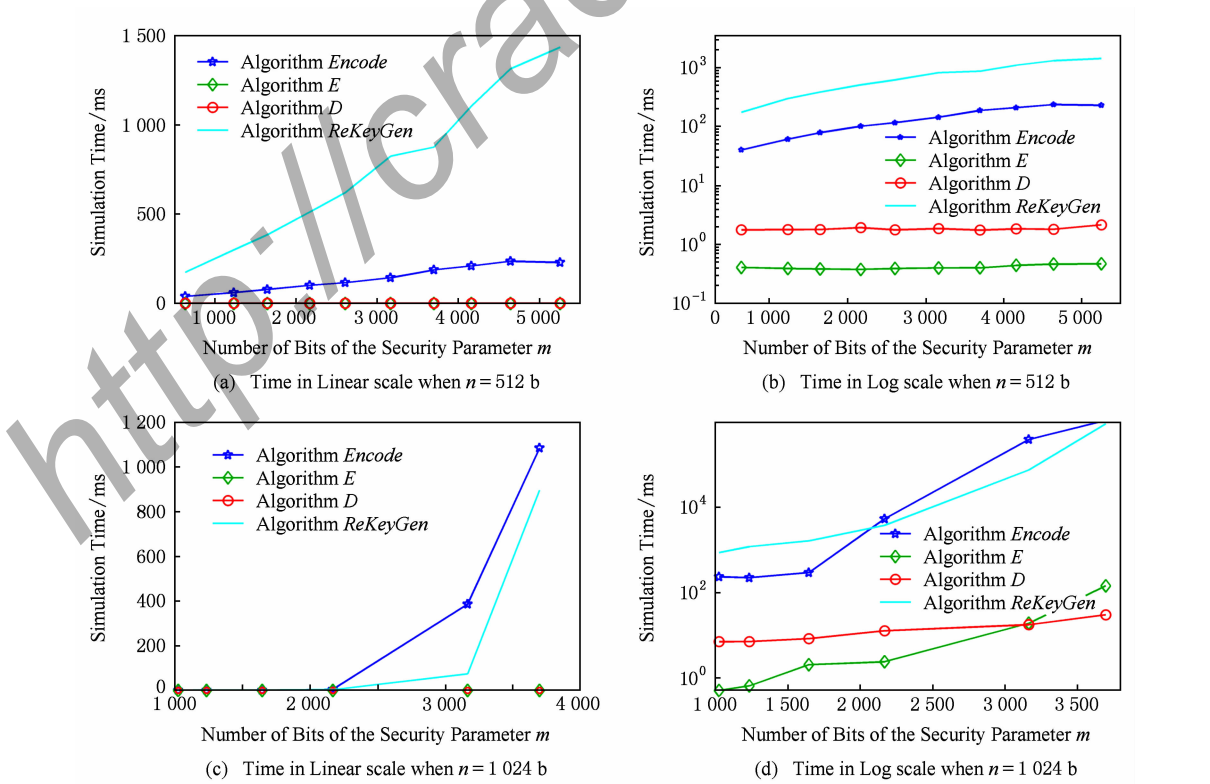


Fig. 6 Simulation of the core algorithms in the MOR primitive

图 6 MOR 核心算法的仿真

本文所提方法和文献[21]的方法,这是因为本文所提方法和文献[21]方法都采用了解密外包的方法.另外,文献[2]中 SK_{u_t} 中的属性数量与 User 端的解密时间成线性关系,而本文所提方法和文献[21]方法将解密运算外包之后,User 端的解密时间与 SK_{u_t} 中的属性数量无关.

6.2 MOR 原语的仿真实验

为了仿真 MOR 原语,采用 FLINT(fast library for number theory),MPFR(multiple precision floating point reliable library,MPIR(multiple precision integer and rational library)实现任意精度运算.其中,LWE 维度参数的位数设为 n 设为 512 b 或 1024 b,高斯分布抽样数目的位数 m 设为关于 n 的多项式 $poly(n)$.算法 $SampleD$ 使用 Nearest-Plane 方法(Gentry[20] 和 Ducas[22]).

图 6(a)(c)中,以线性标尺显示 MOR 原语核心算法的计算代价,图 6(b)(d)则以对数标尺显示.仿真实验结果表明,算法 E 和 D 的运行时间与参数 m 的位数无关,而算法 $ReKeyGen$ 和算法 $Encode$ 的运行时间随着参数 m 位数的增加快速增加.

假设用户可接受的延迟时间临界值是 10^3 ms,那么,当参数 $n=512$ b 时,无论参数 m 设置多大,延迟时间都是用户可以接受的.如果参数 $n=1\,024$ b,则只有参数 m 的位数小于 2 000 b,延迟时间才可以接受.

7 结束语

本文分析了云存储环境中多用户协同编辑访问控制产生的用户状态控制需求,在 Multi-Read-Collaborative-Edit 情景下,提出了一个支持多用户协同编辑的访问控制方法.该方法具有 3 个主要特性:

1) 实现了细粒度访问控制.使得合法用户能够读取/编辑所授权的数据部分,实现了细粒度的访问控制,所设计的云辅助解密方法,减轻了用户端的计算代价,并且该方法在离散对数困难问题假设下被证明没有减弱安全性.

2) 实现了 Data-binding-policy 访问控制方法.Owner 将指定的读权限策略和编辑权限策略绑定在数据上,只有 User 的私钥满足读权限策略时才能读取明文,只有在协同编辑策略匹配成功时才能更新数据.协同编辑控制的复杂性要求 Owner 采用表达能力更丰富的 circuit 定义协同编辑策略,云端根据该策略确定最终修改版本.

3) 实现了云辅助协同编辑策略匹配的不可预测性.本文设计的 MOR 原语具有可重用性和不可预测性,因而基于该原语设计的协同编辑策略匹配方法能够快速判断用户提交的修改数据是否应该接受,并且云不可预测每个用户是否具有写权限.

参 考 文 献

[1] 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 IEEE INFOCOM 2014. Piscataway, NJ: IEEE, 2014: 2013-2021

[2] Hur J, Kang K. Secure data retrieval for decentralized disruption-tolerant military networks [J]. IEEE/ACM Trans on Networking, 2014, 22(1): 16-26

[3] Dong Xin, Yu Jiadi, Zhu Yanmin, et al. SECO: Secure and scalable data collaboration services in cloud computing [J]. Computers and Security, 2015, 50(1): 91-105

[4] Gorbunov S, Vaikuntanathan V, Wee H. Attribute-based encryption for circuits [C] //Proc of the 45th Annual ACM Symp on Theory of Computing (STOC). New York: ACM, 2013: 545-554

[5] Ruj S, Stojmenovic M, Nayak A. Decentralized access control with anonymous authentication of data stored in clouds [J]. IEEE Trans on Parallel and Distributed Systems, 2014, 25(2): 384-394

[6] Li Ming, Yu Shucheng, Zheng Yao, et al. Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption [J]. IEEE Trans on Parallel and Distributed Systems, 2013, 24(1): 131-143

[7] Fan Yanfang, Cai Ying. Collaboration supported mandatory access control model [J]. Journal of Computer Research and Development, 2015, 52 (10): 2411-2421 (in Chinese)
(范艳芳, 蔡英. 支持协作的强制访问控制模型[J]. 计算机研究与发展, 2015, 52 (10): 2411-2421)

[8] Lai Junzuo, Deng R H, Yang Yanjiang, et al. Adaptable ciphertext-policy attribute-based encryption [G] //LNCS 8365: Proc of the Pairing 2013. Berlin: Springer, 2013: 199-214

[9] Chun I F, Huang V S M, Ruan Heming. Arbitrary state attribute-based encryption with dynamic membership [J]. IEEE Trans on Computers, 2014, 63(8): 1951-1961

[10] Rouselakis Y, Waters B. New constructions and proof methods for large universe attribute-based encryption [EB/OL]. 2012[2015-11-05]. http://eprint.iacr.org/2012/583.pdf

[11] Wang Hao, Zheng Zhihua, Wu Lei, et al. Adaptively secure outsourcing ciphertext-policy attribute-based encryption [J]. Journal of Computer Research and Development, 2015, 52 (10): 2270-2280 (in Chinese)

(王皓, 郑志华, 吴磊, 等. 自适应安全的外包 CP-ABE 方案研究[J]. 计算机研究与发展, 2015, 52 (10): 2270-2280)

[12] Boyen X. Attribute-based functional encryption on lattices [G] //LNCS 7785: Proc of TCC 2013. Berlin: Springer, 2013: 122-142

[13] Liu Ximeng, Ma Jianfeng, Xiong Jinbo, et al. Threshold attribute-based encryption with attribute hierarchy for lattices in the standard model [J]. IET Information Security. 2014, 8(4): 217-223

[14] Hohenberger S, Waters B. Online/offline attribute-based encryption [G] //LNCS 8383: Proc of PKC 2014. Berlin: Springer, 2014: 293-310

[15] Garg S, Gentry C, Halevi S, et al. Attribute-based encryption for circuits from multilinear maps [G] //LNCS 8043: Proc of CRYPTO 2013. Berlin: Springer, 2013: 479-499

[16] Xu Jie, Wen Qiaoyan, Li Wenmi, et al. Circuit ciphertext-policy attribute-based hybrid encryption with verifiable delegation in cloud computing [J]. IEEE Trans on Parallel and Distributed Systems, 2016, 27(1): 119-129

[17] Regev O. On lattices, learning with errors, random linear codes, and cryptography [J]. Journal of the ACM, 2009, 56 (6): 34:1-34:40

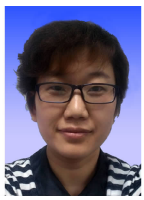
[18] Shi Jiaoli, Huang Chuanghe, Wang Jing, et al. Multi-user collaborative access control scheme in cloud storage [J]. Journal on Communications, 2016, 37 (1): 88-99 (in Chinese)
(史姣丽, 黄传河, 王晶, 等. 云存储下多用户协同访问控制方案[J]. 通信学报, 2016, 37(1): 88-99)

[19] Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption [C] //Proc of IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2007: 321-334

[20] Gentry C, Peikert C, Vaikuntanathan V. Trapdoors for hard lattices and new cryptographic constructions [C] //Proc of the 40th ACM Symp on Theory of Computing (STOC2008). New York: ACM, 2008: 197-206

[21] Yang Kan, Jia Xiaohua, Ren Kui, et al. DAC-MACS: Effective data access control for multi-authority cloud storage systems [C] //Proc of IEEE INFOCOM 2013. Piscataway, NJ: IEEE, 2013: 2895-2903

[22] Ducas L, Lyubashevsky V, Prest T. Efficient identity-based encryption over NTRU lattices [G] //LNCS 8874: Proc of ASIACRYPT 2014. Berlin: Springer, 2014: 22-41

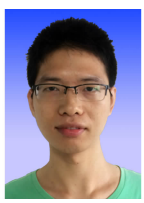


Shi Jiaoli, born in 1979. PhD candidate, associate professor. Member of CCF. Her main research interests include network security and CDN.

Huang Chuanhe, born in 1963. PhD, professor, PhD supervisor. Senior member of CCF. His main research interests include network security, WDM networks, mobile ad hoc networks, CDN, and quantum computing.

He Kai, born in 1987. PhD. His main research interests include network security and cryptography.

Shen Xieyang, born in 1988. PhD candidate. His main research interests include network security and cryptography.



Hua Chao, born in 1992. Master. His main research interest is long distance wireless network.