基于直接匿名证明的 *k* 次属性认证方案

柳欣[1,2]，徐秋亮[3]，张斌[1,2]，张波[4]

（1. 山东青年政治学院信息工程学院，山东 济南 250103；
2. 山东省高校信息安全与智能控制重点实验室（山东青年政治学院），山东 济南 250103；
3. 山东大学软件学院，山东 济南 250101；4. 济南大学信息科学与工程学院，山东 济南 250022）

摘 要：当前，已有 *k* 次属性认证（简称 *k*-TABA）方案以及相关属性认证方案的主要缺点是认证子协议的运算复杂度依赖于属性认证策略的规模，而且并未考虑成员废除和属性更新问题。基于直接匿名证明、集合成员身份证明和密文策略属性加密技术构造了新的 *k*-TABA 方案。为了进一步优化用户端运算效率，首先对底层属性加密方案进行修改，然后利用 Green 等的密钥绑定技术对解密过程进行外包。该方案不但可部署于可信平台，而且支持可表述性认证策略。此外，该方案满足多个理想性质，诸如注册过程可验证性、成员废除和属性更新等。该方案最显著的性能优势是用户在认证阶段的运算开销为常数。

关键词：属性认证；直接匿名证明；密文策略属性加密；线性秘密分享；外包解密
中图分类号：TN918.2
文献标识码：A
doi: 10.11959/j.issn.1000-436x.2018279

k-times attribute-based authentication scheme using direct anonymous attestation

LIU Xin[1,2], XU Qiuliang[3], ZHANG Bin[1,2], ZHANG Bo[4]

1. School of Information Engineering, Shandong Youth University of Political Science, Ji'nan 250013, China
2. Key Laboratory of Information Security and Intelligent Control in Universities of Shandong (Shandong Youth University of Political Science), Ji'nan 250103, China
3. Software College, Shandong University, Ji'nan 250101, China
4. School of Information Science and Engineering, University of Ji'nan, Ji'nan 250022, China

Abstracts: At present, the main drawbacks of existing *k*-times attribute-based authentication (abbreviated to *k*-TABA) schemes and related attribute-based authentication schemes are that the computation cost of the authentication process depends on the size of the access formula and none of these schemes considers the problems of member revocation and attribute update. A new *k*-TABA scheme was constructed based on the building blocks of direct anonymous attestation, set membership proof and ciphertext-policy attribute-based encryption. Moreover, in order to reduce user's calculation as much as possible, the underlying attribute-based encryption scheme was modified, and then the main decryption operations were outsourced by using the key binding technique of Green et al. The new scheme can be deployed on a trusted platform and support expressive authentication policies. In addition, it also satisfies several ideal properties, such as registration process verifiability, member revocation, attribute update, and so on. The significant performance advantage of the new scheme is that the computation overhead of the user in the authentication phase is constant.

Key words: attribute-based authentication, direct anonymous attestation, ciphertext-policy attribute-based encryption, linear secret sharing, outsourced decryption

收稿日期：2017−09−25；**修回日期**：2018−09−10

通信作者：柳欣，lxonne@163.com

基金项目：国家自然科学基金资助项目（No. 61173139）；山东省自然科学基金资助项目（No.ZR2015FL023, No.ZR2014FL011, No. ZR2015FL022）；山东省高等学校科学技术计划基金资助项目（No.J17KA081, No.J15LN16, No.J13LN23）；山东青年政治学院博士科研启动经费资助项目（No.14A007）

Foundation Items: The National Natural Science Foundation of China (No.61173139), Shandong Provincial Natural Science Foundation (No.ZR2015FL023, No.ZR2014FL011, No.ZR2015FL022), The Project of Shandong Province Higher Educational Science and Technology Program (No.J17KA081, No.J15LN16, No.J13LN23), The Doctoral Research Start-up Funding Project of Shandong Youth University of Political Science (No.14A007)

1 引言

当前，云服务模式因其在运算能力强、存储空间无限、容易扩展和按需提供服务[1]等方面的优势而逐渐赢得企业和个人用户的青睐。在各类云应用系统（如分布式医疗系统[2-3]、企业计算[4]等）的设计中，需要着重考虑用户的隐私保护和细粒度访问控制问题。对此，属性认证技术提供了较好的解决方案。在属性认证方案中，认证过程通常由用户对某项服务的请求触发。一旦服务提供商（SP, service provider）接收到该项请求，就会返回认证所需的属性要求（或称为访问结构）。此时，用户向 SP 提供自己掌握这些属性的证明。若证明有效，用户将得到后者的访问授权。在整个认证过程中，用户不需要揭示自己具体使用的属性集合。相对于传统的基于公钥的认证[5-7]，属性认证的最大优势是允许系统管理者根据用户属性为用户群体中的每个成员定制访问权限，使不同用户可以利用属性私钥访问不同的敏感数据或服务。

在文献[3]中，Zhou 等提出一个病人自我可控的隐私保护属性认证方案。该方案定义了 3 个隐私保护等级。1）得到病人直接授权的医生既能访问病人健康信息，也能实现对病人的身份认证。2）得到病人间接授权的医生和研究机构仅能访问病人健康信息。3）未得到授权的人将无法获得病人的健康信息。然而，Zhou 等的方案仅支持 d-out-of-n 门限认证策略。2014 年，Lian 等[8]指出在许多情况下，认证策略并非总是静态的。为此，提出一个支持动态认证策略的属性认证方案。该方案可视为 Maji 等[9]属性签名（ABS, attribute-based signatures）方案的直接应用。此后，Li 等[4]指出某些应用要求将用户集合划分为多个不同的小组，同时为组内用户分配不同的属性。在认证过程中，要求同组的多个成员对各自的属性私钥进行合并。为此，通过对 Maji 等[9]ABS 方案做出扩展，Li 等提出一个适合于企业计算环境的合作属性认证方案。2015 年，Yang 等[10]提出基于一次性属性树的属性认证方案，但缺点是：1) 用户群体是采用静态方式建立的，因此不允许动态增加新的用户；2) 属性权威（AA，attribute authority）有能力从认证协议副本中恢复用户的身份，因此用户必须完全信任 AA。此外，Yang 等[11]还提出一个基于动态属性树的属性认证方案。然而，该方案与文献[10]方案具有相同的缺点。此外，用户必须在认证过程中明确地向 SP 揭示自己使用的属性子集，即不满足属性匿名性。在文献[12]中，Li 等指出在实际应用中有必要利用分布式 AA 取代单一的中央权威，并且提出一个多权威属性认证方案。然而，该方案的缺点是：1) 仅支持 d-out-of-n 门限认证策略；2) 在密钥发布阶段，用户总共需要执行 $N(N-1)$ 次两方密钥协商子协议，其中，N 表示系统中的 AA 数量；3) 当需要增加或删除某个 AA 时，所有剩余的 AA 需要重新执行系统建立协议，而且用户需要分别与每个 AA 重新执行低效的密钥协商子协议。

最近，Yuen 等[13]提出第一个 k 次属性访问控制方案模型以及一个具体方案。Yuen 等指出，此类方案可以同时解决按次付费的云服务模式关于“灵活的匿名访问控制”和“用户访问次数受限”的应用需求。然而，该方案的缺点如下所示。1) 该方案是在标准 PC 平台上设计的。一旦 PC 被攻破，该用户即被完全攻破，从而无法体现其安全模型中关于“用户部分被攻破”的情况。2) 在注册协议中，用户无法对 AA 提供的属性私钥进行验证，即用户必须完全信任 AA。3) Yuen 等指出，为了防止恶意 AA 进行陷害，用户需要在注册协议中选取用户私钥。然而，该方案的安全模型定义却遗漏了可开脱性。4) 在认证过程中，用户与 SP 的运算量线性依赖于访问结构的规模。5) 为了实施对访问次数的控制，要求用户在认证期间向 SP 提供已执行的认证次数。然而，这种方式在某些情况下不满足无关联性。尽管 Yuen 等指出可以通过引入区间证明技术加以改进，但并未提供具体方案。

本文认为，Yuen 等的方案可视为一种 k 次属性认证（k-TABA, k-times attribute-based authentication）方案。相对于基于传统公钥认证技术的 k-TAA（k-times anonymous authentication）方案[5-7]，k-TABA 方案因支持细粒度访问控制而更适合于当前的云应用环境。本文的研究目标是设计认证过程更为高效的 k-TABA 方案，同时克服已有属性认证方案[3-4,8,10-13]的诸多缺点，主要包括：1) 用户群体以静态方式建立，且用户无法在注册过程中对属性私钥进行验证；2) 用户在认证阶段的运算耗费与底层属性认证策略满足线性依赖关系；3) 既不允许 AA 对泄露的用户私钥执行废除，也不允许用户申请更新特定属性值。为此，本文基于可信计算领域的直接匿名证明[14-18]、Zhang 等[19]的密文策略属性

加密方案和区间证明技术[20]构造了新的 k-TABA 方案。需要指出的是，Zhang 等的方案是采用属性树方式描述的。本文方案采用了基于线性秘密分享方案的描述方式，从而能更好地与 Green 等[21]的密钥绑定技术相结合。借助此项技术，本文提出 Zhang 等方案的外包解密版本，并将其安全性由 CPA（chosen plaintext attack）安全性增强至 RCCA（replayable chosen-ciphertext attack）安全性。此外，本文对 Chen[15]的直接匿名证明安全模型进行扩展，从而得出新的 k-TABA 方案安全模型。同时，在该模型下为新方案提供了安全性证明。与 Yuen 等的 k-TABA 方案相比，本文方案具备以下优势。1) 用户端运算由主机平台（Host）和可信的 TPM（trusted platform module）芯片协同完成，使得即使敌手控制了 Host，仍然无法以用户的身份通过认证过程。2) 引入了直接匿名证明方案的验证者本地废除和 Zhang 等的属性更新机制，从而满足了现实应用中对泄露的用户私钥实施废除和允许用户申请更新特定属性值的需要。3)在用户注册过程中，允许用户对 AA 提供的属性私钥进行验证，从而无需假设 AA 完全可信。4) 在认证阶段，允许用户借助云服务器完成复杂的属性解密运算，而且用户端 TPM 芯片与 Host 的绝大部分运算任务均能预先以离线方式执行。于是，用户端无需执行双线性对运算（简称为对运算）且运算开销独立于用户属性集合或访问结构。作为总结，本文的主要贡献体现在以下方面。1) 克服了上述已有 k-TABA 方案的多个缺陷，并提出改进的安全模型。2) 结合直接匿名证明技术增强了用户端的安全性。3) 结合无需执行对运算的知识证明、预计算和外包解密等技术在最大程度上优化了用户在认证、属性更新阶段的运算效率。4) 去除了已有方案中关于“AA 在注册阶段完全可信”的较强假设。

2　预备知识

本文使用了对称的双线性群环境 $(G_0, G_T, p, \hat{e})$，其中，G_0, G_T 表示素数 p 阶循环群，且 $\hat{e}$ 表示双线性映射，使 $\hat{e}: G_0 \times G_0 \to G_T$。

2.1　安全假设

q-SDH（the q-strong Diffie-Hellman）假设[6]：对于任何的概率多项式时间（PPT, probabilistic polynomial time）算法 $\mathfrak{A}$，以下的概率是可忽略的，即

$$Adv_{\mathfrak{A}}^{q\text{-SDH}}(\lambda) = \Pr[\mathfrak{A}(\boldsymbol{t}, P, P^{\gamma}, P^{\gamma^2}, \cdots, P^{\gamma^q}) = (c, P^{\frac{1}{\gamma+c}}) \wedge (c \in \mathbb{Z}_p)]$$

其中，$\boldsymbol{t} = (G_0, G_T, p, \hat{e}), P \in G_0, \gamma \in \mathbb{Z}_p^*$。

DBDH（decisional bilinear Diffie-Hellman）假设[14]：对于任何的 PPT 算法 $\mathfrak{A}$，以下的概率是可忽略的，即

$$Adv_{\mathfrak{A}}^{\text{DBDH}}(\lambda) = |\Pr[\mathfrak{A}(P, P^a, P^b, P^{\varsigma}, \hat{e}(P,P)^{ab\varsigma}) = 1] - \Pr[\mathfrak{A}(P, P^a, P^b, P^{\varsigma}, \hat{e}(P,P)^{d}) = 1]|$$

其中，$P \in G_0, a, b, \varsigma, d \in \mathbb{Z}_p^*$。

DBDHI（decisional bilinear Diffie-Hellman inversion）假设[6]：对于任何的 PPT 算法 $\mathfrak{A}$，以下的概率是可忽略的，即

$$Adv_{\mathfrak{A}}^{\text{DBDHI}}(\lambda) = |\Pr[\mathfrak{A}(\boldsymbol{t}, P, P^{\gamma}, P^{\gamma^2}, \cdots, P^{\gamma^q}, \hat{e}(P,P)^{\frac{1}{\gamma}}) = 1] - \Pr[\mathfrak{A}(\boldsymbol{t}, P, P^{\gamma}, P^{\gamma^2}, \cdots, P^{\gamma^q}, R) = 1]|$$

其中，$\boldsymbol{t} = (G_0, G_T, p, \hat{e}), P \in G_0, \gamma \in \mathbb{Z}_p^*, R \in G_T^*$。

2.2　基于集合成员身份的区间证明技术

首先，权威产生 Boneh-Boyen 签名方案[22]的密钥对 $(w = g^{\gamma}, \gamma)$。对于集合 $\Phi = \{1, 2, \cdots, \overline{n}\}$ 中的每个元素 k，该权威计算 $o_k = g^{\frac{1}{\gamma+k}}$。同时，公开集合 $\Sigma = \{o_1, o_2, \cdots, o_n\} = \{g^{\frac{1}{\gamma+1}}, g^{\frac{1}{\gamma+2}}, \cdots, g^{\frac{1}{\gamma+n}}\}$。

现在，用户构造知识证明 $\pi = PK\{(k): k \in \Phi\}$ 等价于构造证明 $\pi = PK\{(o_k): o_k = g^{\frac{1}{\gamma+k}} \wedge o_k \in \Sigma\}$。文献[20]提出了避免用户执行对运算的技术，即用户选取 $l, \nu \in \mathbb{Z}_p$，设置 $\tilde{o}_k = o_k^l$，并将 π 转换为 $\pi = PK\{(k, \nu, l): com = g^k h^{\nu} \wedge E = \tilde{o}_k^{-k} g^l\}$ 的形式。此外，验证者需要额外检查是否满足 $\hat{e}(E, g) = \hat{e}(\tilde{o}_k, w)$。

2.3　访问结构与线性秘密分享方案

令 $\mathfrak{P} = \{P_1, P_2, \cdots, P_n\}$ 表示参与方集合。本文称集合 $\mathbb{A} \subseteq 2^{\{P_1, P_2, \cdots, P_n\}}$ 是单调的，条件是对于任何的集合 B, C，若 $B \in \mathbb{A}$ 且 $B \subseteq C$，则满足 $C \in \mathbb{A}$。本文称 $\mathfrak{P}$ 的（单调的）非空子集 $\mathbb{A}$ 为（单调的）访问结构，即 $\mathbb{A} \subseteq 2^{\{P_1, P_2, \cdots, P_n\}} \setminus \{\varnothing\}$。$\mathbb{A}$ 中的集合称为授权集合，且不属于 $\mathbb{A}$ 的集合称为非授权集合[21]。

本文称在 $\mathbb{Z}_p$ 上构造的秘密分享方案 Π 是关于 $\mathfrak{P}$ 的线性秘密分享方案（LSSS, linear secret sharing schemes）[21]，条件是以下性质同时得到满足：1) 参与方的秘密份额构成了 $\mathbb{Z}_p$ 上的秘密向量；2) 存在 l

行 $\bar{n}$ 列矩阵 $\boldsymbol{M}$，且该矩阵称为方案 Π 的份额产生矩阵。存在函数 ρ，该函数能将矩阵 $\boldsymbol{M}$ 的第 i 行 $\boldsymbol{M}_i$ 映射到参与方 $\rho(i)$。定义列向量 $\vec{\boldsymbol{v}} = (s, r_2, r_3, \cdots, r_n)$，其中，$s \in \mathbb{Z}_p$ 表示待分享的秘密元素，$r_2, r_3, \cdots, r_n \in \mathbb{Z}_p$ 为随机元素，则称 $\lambda_i = \boldsymbol{M}_i \vec{\boldsymbol{v}}$ 构成由参与方 $\rho(i)$ 掌握的关于元素 s 的秘密份额。

2.4 支持外包解密的密文策略属性加密方案及其安全性

属性加密（ABE, attribute-based encryption）方案是属性密码学的一个重要分支，具体可细分为密钥策略属性加密（KP-ABE, key-policy ABE）方案和密文策略属性加密（CP-ABE, ciphertext-policy ABE）方案[2]。相对于标准的 ABE 方案，支持外包解密的 ABE 方案需要额外定义 3 个算法，即 $KeyGen_{\text{out}}, Transform_{\text{out}}, Decrypt_{\text{out}}$ [21]。其中，$KeyGen_{\text{out}}$ 算法的作用是产生用户的解密私钥 ASK 和转换钥 TK；$Transform_{\text{out}}$ 算法的作用是利用 TK 对密文 CT 执行解密，得到部分解密的结果 CT'；$Decrypt_{\text{out}}$ 算法的作用是利用 ASK 将 CT' 转换为真正的明文 m。

文献[21, 23]指出，支持外包解密的 ABE 方案应当满足 RCCA 安全性和可验证性。其中，RCCA 安全性是一个强度介于 CCA（chosen-ciphertext attack）安全性和 CPA 安全性之间的概念。在本质上，RCCA 安全性允许在“不以有意义方式改变消息内容”的条件下对密文做出修改[21]。本文着重考虑支持外包解密的 CP-ABE 方案的安全性。具体地，本文称支持外包解密的 CP-ABE 方案满足选择性的 RCCA 安全性[21]，条件是任何的 PPT 算法 $\mathfrak{A}$ 都无法以不可忽略的概率在以下的实验中获胜：在初始化阶段，敌手 $\mathfrak{A}$ 向模拟器 $\mathfrak{S}$ 发送用于产生挑战密文的访问结构 $\mathbb{A}^*$。在系统建立阶段，$\mathfrak{S}$ 向 $\mathfrak{A}$ 提供 APK。在询问 1 阶段，允许 $\mathfrak{A}$ 提出预言询问 $Create(S), Corrupt(i), Decrypt(i, Cha)$。其中，预言机 $Create(\cdot)$ 产生关于属性集合 S 的解密私钥 ASK 和转换钥 TK，并且返回 TK。预言机 $Corrupt(\cdot)$ 返回第 i 次 $Create$ 询问中产生的 ASK。预言机 $Decrypt(\cdot,\cdot)$ 利用第 i 次 $Create$ 询问中产生的 ASK 解密密文 Cha，并且返回解密结果 m。在挑战阶段，$\mathfrak{A}$ 输出消息 M_0, M_1，条件是 $\mathfrak{A}$ 并不掌握能满足访问结构 $\mathbb{A}^*$ 的解密私钥。此时，$\mathfrak{S}$ 向 $\mathfrak{A}$ 返回利用 $\mathbb{A}^*$ 产生的关于消息 $M_{\bar{b}}(\bar{b} \in \{0,1\})$ 的挑战密文 Cha^*。在询问 2 阶段，允许 $\mathfrak{A}$ 继续提出上述类型的询问。但是，不允许 $\mathfrak{A}$ 通过 $Corrupt$ 询问获得能满足 $\mathbb{A}^*$ 的解密私钥，也不允许 $\mathfrak{A}$ 直接将 Cha^* 作为 $Decrypt$ 询问的内容。最终，若 $\mathfrak{A}$ 能对秘密比特 $\bar{b}$ 做出正确猜测，则判定它在实验中获胜。

可验证性是指用户可以验证由服务器执行的密文转换过程是否正确[23]。可验证性实验与上述实验的区别如下：1) 去除初始化阶段；2) 在询问 1 和询问 2 阶段，同样允许 $\mathfrak{A}$ 提出预言询问 $Create(S), Corrupt(i), Decrypt(i, Cha)$，但不对 $\mathfrak{A}$ 的询问内容做出限制；3) 在挑战阶段，$\mathfrak{A}$ 输出 $M^*, \mathbb{A}$。此时，$\mathfrak{S}$ 返回 $Cha^* = Encrypt(APK, M^*, \mathbb{A})$；4) 在最终的输出阶段，$\mathfrak{A}$ 输出属性集合 S^* 和部分解密结果 $Cha^{*\prime}$。若满足 $Decrypt_{\text{out}}(APK, Cha^*, Cha^{*\prime}, ASK_{S^*}) \notin \{M^*, \perp\}$，则判定 $\mathfrak{A}$ 在实验中获胜。

2.5 直接匿名证明

直接匿名证明（DAA, direct anonymous attestation）[14-18]是一类可用于实现可信平台模块远程证明的匿名签名方案，且所得签名可以确保用户的隐私。在 DAA 方案中，签名者角色可划分为主要签名者和辅助签名者。前者由运算能力受限的 TPM 芯片充当，后者则由具有冗余计算能力但安全等级更低的 Host 充当。由于 TPM 掌握签名私钥 f，因此 Host 无法在 TPM 不参与的情况下独立产生签名。最新研究表明，许多 DAA 方案中的 TPM 可被作为静态 Diffie-Hellman 预言机使用，即给定群元素 B，TPM 输出 B^f。于是，若 Host 被攻破，敌手可以利用 Brown-Gallant 算法提取出私钥 f [17-18]。因此，在 DAA 方案的设计与应用中，必须破坏静态 Diffie-Hellman 预言机的存在条件，从而确保方案的基础安全性。

3 k 次属性认证方案的语法定义与安全模型

首先，本文定义的 k-TABA 方案包含以下参与方，即用户 $User$、属性权威 AA 和服务提供商 SP，其中，$User$ 可划分为 TPM 和 Host。此类方案由以下的算法/协议构成。

$TSetup$ 这是一个可信的系统建立算法，用于产生方案的系统参数 TPK。

$ASetup$ 该算法由 AA 执行，用于产生方案的公钥 APK 和 AA 的主密钥 MSK。

$USetup$ 该算法由 $User$ 执行，用于产生密钥对 $(upk, usk), (hpk, hsk)$。

AttGen 这是由 *User* 与 *AA* 执行的协议。通过该协议，*User* 获得 *AA* 提供的伪名 *nym*，成员证书 *cre* 以及关于属性集合 *S* 的属性私钥 *ASK*。

Auth 这是由 *User* 与 *SP* 共同执行的属性认证协议。

AUpdate 当特定属性值发生改变（如由 j 更新为 w），*User* 可以通过该协议向 *AA* 申请更新钥 $UK_{j\to w}$，从而对 *ASK* 中对应于属性 j 的部分进行更新。同时，*AA* 为其他拥有该属性且并未执行属性更新的用户提供更新钥 UK'_j。

在本文安全模型下，安全的 k-TABA 方案应当同时满足以下性质。

正确性：假设 *TPK* 是利用 *TSetup* 算法产生的，(MSK, APK) 是利用 *ASetup* 算法产生的，$(upk, usk), (hpk, hsk)$ 是诚实用户 *User* 利用 *USetup* 算法产生的。*User* 通过与 *AA* 执行 *AttGen* 协议而获得伪名 *nym*，成员证书 *cre* 以及关于属性集合 *S* 的属性私钥 *ASK*。当 *User* 利用 (nym, cre, ASK) 与诚实的 *SP* 执行 *Auth* 协议时，只要 *S* 满足 *SP* 在底层 CP-ABE 方案密文 *Cha* 中嵌入的访问结构 $\mathbb{A}$（此后标记为 $S \models \mathbb{A}$）且已执行的认证次数 *count* 不超过上界 n，则 *User* 必然能通过此次认证过程。此外，当自己的特定属性值由 j 更新为 w 时，*User* 可以通过与 *AA* 执行 *AUpdate* 协议而获得更新后的属性私钥 *ASK′*。

隐私性：若 *User* 的认证次数 *count* 不超过上界 n，则包括 *AA* 与 *SP* 在内的任何参与方都无法对该用户执行 *Auth* 协议的过程进行关联。另外，对于 2 个拥有相同属性集合 *S* 的用户 $User_1, User_2$，包括 *AA* 与 *SP* 在内的任何参与方都无法对他们执行 *Auth* 协议的过程进行关联。

可靠性：若 *User* 的认证次数上界为 n，则它无法与 *SP* 成功地执行 $n+1$ 次 *Auth* 协议而不被发觉。

抗合谋攻击：对于每个被攻破的用户，在其属性集合 *S* 并不满足给定访问结构 $\mathbb{A}$（此后标记为 $S \not\models \mathbb{A}$）的情况下，即使他们联合起来，也无法与 *SP* 成功地执行 *Auth* 协议。

可追踪性：假设 *User* 拥有能满足给定访问结构 $\mathbb{A}$ 的属性集合，即使敌手能攻破 *User* 的主机平台，也无法在绕开 TPM 的情况下独立执行 *Auth* 协议，或实现对 *User* 的陷害。

3.1　隐私性实验

敌手 $\mathfrak{A}$ 与模拟器 $\mathfrak{S}$ 共同执行以下过程。

初始化　$\mathfrak{S}$ 执行 *TSetup* 算法，产生 *TPK*。$\mathfrak{S}$ 执行 *ASetup* 算法，产生 MSK, APK。$\mathfrak{S}$ 初始化列表 $\mathfrak{L}$，并且向 $\mathfrak{A}$ 发送 TPK, APK, MSK。

询问 1　允许 $\mathfrak{A}$ 向 $\mathfrak{S}$ 提出以下类型的预言询问。

-*USetup*()　$\mathfrak{S}$ 产生 $(usk, upk), (hsk, hpk)$，执行 $\mathfrak{L} \leftarrow \mathfrak{L} \cup (upk, usk, hpk, hsk, *, *, *, *, *, *)$（* 表示未知元素），并且返回 upk, hpk。

-*AttGen*(S, upk)　$\mathfrak{S}$ 以用户 *upk* 的身份与 $\mathfrak{A}$ 执行 *AttGen* 协议，并且获得后者产生的 *nym*, *cre* 和 *ASK*。最终，$\mathfrak{S}$ 将 $\mathfrak{L}$ 中的表目 $(upk, usk, hpk, hsk, *, *, *, *, *, *)$ 更新为 $(upk, usk, hpk, hsk, nym, cre, ASK, S, c=0, count=0)$，其中，$c=0$ 表示未攻破，$c=1$ 表示已攻破，且 $count=0$ 表示已执行的认证次数。

-*Corrupt*(nym)　$\mathfrak{S}$ 根据伪名 *nym* 在 $\mathfrak{L}$ 中找到表目 $(upk, usk, hpk, hsk, nym, cre, ASK, S, c, count)$。若 $c=0$，则返回 usk, hsk，并且将该表目更新为 $(upk, usk, hpk, hsk, nym, cre, ASK, S, c=1, count)$。

-*Auth*(Cha, nym)　$\mathfrak{S}$ 根据 *nym* 在 $\mathfrak{L}$ 中找到表目 $(upk, usk, hpk, hsk, nym, cre, ASK, S, c, count)$。假设 $\mathbb{A}$ 表示嵌入在底层 CP-ABE 方案密文 *Cha* 中的访问结构。若 $S \models \mathbb{A}$ 且 $count < n$，则 $\mathfrak{S}$ 以用户 *nym* 的身份与 $\mathfrak{A}$ 执行 *Auth* 协议。最终，$\mathfrak{S}$ 将该表目更新为 $(upk, usk, hpk, hsk, nym, cre, ASK, S, c, count+1)$。

挑战　$\mathfrak{A}$ 输出 $(nym_0^*, nym_1^*, Cha^*)$。假设 $\mathbb{A}^*$ 表示嵌入在底层 CP-ABE 方案密文 Cha^* 中的访问结构。$\mathfrak{S}$ 检查是否满足 $(upk_0^*, usk_0^*, hpk_0^*, hsk_0^*, nym_0^*, cre_0^*, ASK_0^*, S_0^*, c_0^*, count^*), (upk_1^*, usk_1^*, hpk_1^*, hsk_1^*, nym_1^*, cre_1^*, ASK_1^*, S_1^*, c_1^*, count^*) \in \mathfrak{L}, S_0^* \models \mathbb{A}^*, S_1^* \models \mathbb{A}^*, c_0^* = c_1^* = 0$ 且 $count^* < n$。若是，$\mathfrak{S}$ 选取 $\overline{b} \in_R \{0,1\}$，并且以用户 $nym_{\overline{b}}^*$ 的身份与 $\mathfrak{A}$ 执行 *Auth* 协议。在该协议结束后，$\mathfrak{S}$ 将 $\mathfrak{L}$ 中的表目 $(upk_{\overline{b}}^*, usk_{\overline{b}}^*, hpk_{\overline{b}}^*, hsk_{\overline{b}}^*, nym_{\overline{b}}^*, cre_{\overline{b}}^*, ASK_{\overline{b}}^*, S_{\overline{b}}^*, c_{\overline{b}}^*, count^*)$ 更新为 $(upk_{\overline{b}}^*, usk_{\overline{b}}^*, hpk_{\overline{b}}^*, hsk_{\overline{b}}^*, nym_{\overline{b}}^*, cre_{\overline{b}}^*, ASK_{\overline{b}}^*, S_{\overline{b}}^*, c_{\overline{b}}^*, count^*+1)$，将表目 $(upk_{1-\overline{b}}^*, usk_{1-\overline{b}}^*, hpk_{1-\overline{b}}^*, hsk_{1-\overline{b}}^*, nym_{1-\overline{b}}^*, cre_{1-\overline{b}}^*, ASK_{1-\overline{b}}^*, S_{1-\overline{b}}^*, c_{1-\overline{b}}^*, count^*)$ 更新为 $(upk_{1-\overline{b}}^*, usk_{1-\overline{b}}^*, hpk_{1-\overline{b}}^*, hsk_{1-\overline{b}}^*, nym_{1-\overline{b}}^*, cre_{1-\overline{b}}^*, S_{1-\overline{b}}^*, c_{1-\overline{b}}^*, count^*+1)$。

询问 2　允许 $\mathfrak{A}$ 继续提出询问 1 阶段中的各类询问，但不允许它提出询问 $Corrupt(nym_0^*)$ 或 $Corrupt(nym_1^*)$。

猜测　最终，$\mathfrak{A}$ 输出对 $\overline{b}$ 的猜测结果 $\overline{b}'$。若

$\overline{b}=\overline{b}'$，则判定 $\mathfrak{A}$ 在实验中获胜。

3.2 可靠性实验

$\mathfrak{A}$ 与 $\mathfrak{S}$ 共同执行以下过程。

初始化 $\mathfrak{S}$ 执行 *TSetup* 算法，产生 *TPK*。同时，$\mathfrak{S}$ 运行 *ASetup* 算法，产生 APK, MSK。$\mathfrak{S}$ 初始化列表 $\mathfrak{L}$，并且向 $\mathfrak{A}$ 发送 TPK, APK。

询问 允许 $\mathfrak{A}$ 向 $\mathfrak{S}$ 提出以下类型的预言询问。

-*USetup*() 同隐私性实验。

-*AttGen*(S, upk) $\mathfrak{S}$ 以诚实 *AA* 的身份与 $\mathfrak{A}$ 共同执行 *AttGen* 协议，并且为 $\mathfrak{A}$ 产生 nym, cre, ASK。

-*Corrupt*(nym) $\mathfrak{S}$ 根据伪名 *nym* 在 $\mathfrak{L}$ 中找到表目 $(upk, usk, hpk, hsk, nym, cre, ASK, S, c, count)$。若 $c=0$，则返回 usk, hsk，并且将该表目更新为 $(upk, usk, hpk, hsk, nym, cre, ASK, S, c=1, count)$。

-*Auth*(Cha, nym) 同隐私性实验。

挑战：$\mathfrak{A}$ 输出 $\mathbb{A}^*, nym^*$。$\mathfrak{S}$ 检查 nym^* 是否是借助 *AttGen* 询问产生的，若是，则 $\mathfrak{S}$ 以诚实 *SP* 的身份与 $\mathfrak{A}$ 执行 $n+1$ 次 *Auth* 协议，并且利用 $\mathbb{A}^*$ 产生底层 CP-ABE 方案的挑战密文 *Cha*。最终，若这 $n+1$ 次执行过程都获得成功，则判定 $\mathfrak{A}$ 在实验中获胜。

3.3 抗合谋攻击实验

$\mathfrak{A}$ 与 $\mathfrak{S}$ 共同执行以下过程。

初始化 $\mathfrak{A}$ 向 $\mathfrak{S}$ 提供访问结构 $\mathbb{A}^*$，$\mathfrak{S}$ 运行 *TSetup* 算法，产生 *TPK*。$\mathfrak{S}$ 运行 *ASetup* 算法，产生 APK, MSK。$\mathfrak{S}$ 初始化列表 $\mathfrak{L}, \mathfrak{L}_{nym}$，并且向 $\mathfrak{A}$ 发送 TPK, APK。

询问 1 允许 $\mathfrak{A}$ 向 $\mathfrak{S}$ 提出以下类型的预言询问。

-*USetup*() 同隐私性实验。

-*AttGen*(S, upk) 同可靠性实验，唯一的限制是要求 $S|\neq \mathbb{A}^*$。此外，$\mathfrak{S}$ 执行 $\mathfrak{L}_{nym} \leftarrow \mathfrak{L}_{nym} \cup (nym)$。

-*AUpdate*(S, j, w) $\mathfrak{S}$ 为 $\mathfrak{A}$ 产生更新钥 $UK_{j\to w}$，唯一的限制是 $S'|\neq \mathbb{A}^*$，其中 S' 表示对 S 执行属性更新的结果。

-*Corrupt*(nym) 同可靠性实验。

-*Auth*(Cha, nym) 同隐私性实验。

挑战 $\mathfrak{A}$ 输出 K_{SP_0}, K_{SP_1}, NYM，其中，*NYM* 表示由被废除用户的伪名构成的集合。$\mathfrak{S}$ 检查是否满足 $NYM \subseteq \mathfrak{L}_{nym}$，若是，则选取 $\overline{b} \in_{\mathrm{R}} \{0,1\}$，并且以诚实 *SP* 的身份与 $\mathfrak{A}$ 执行 *Auth* 协议。

在该协议的执行过程中，$\mathfrak{S}$ 利用 $\mathbb{A}^*$ 产生关于 $K_{SP_{\overline{b}}}$ 的底层 CP-ABE 方案的挑战密文 Cha^*。

询问 2 允许 $\mathfrak{A}$ 继续提出询问 1 阶段中的各类询问，但不允许它利用 *AUpdate* 询问获得可以对 Cha^* 执行解密的属性私钥。同时，对于 *AttGen* 询问，要求所产生的 *nym* 满足 $nym \notin NYM$。

猜测 最终，$\mathfrak{A}$ 输出对 $\overline{b}$ 的猜测结果 $\overline{b}'$。若 $\overline{b}=\overline{b}'$，则判定 $\mathfrak{A}$ 在实验中获胜。

3.4 可追踪性实验

该实验的执行过程分以下 2 个模式。

模式 1

初始化 $\mathfrak{S}$ 执行 *TSetup* 算法，产生 *TPK*。同时，$\mathfrak{S}$ 运行 *ASetup* 算法，产生 APK, MSK。$\mathfrak{S}$ 初始化列表 $\mathfrak{L}$，并且向 $\mathfrak{A}$ 发送 TPK, APK。

询问 允许 $\mathfrak{A}$ 向 $\mathfrak{S}$ 提出以下类型的预言询问。

-*USetup*() 对该询问的处理分以下 2 种情况。在情况 1 中，$\mathfrak{S}$ 为用户产生 $(upk, usk), (hpk, hsk)$，执行 $\mathfrak{L} \leftarrow \mathfrak{L} \cup (upk, usk, hpk, hsk, *, *, *, *, c=0, *)$，并且返回 upk, hpk。在情况 2 中，$\mathfrak{S}$ 接收由 $\mathfrak{A}$ 产生的 $(upk, usk), (hpk, hsk)$，并且执行 $\mathfrak{L} \leftarrow \mathfrak{L} \cup (upk, usk, hpk, hsk, *, *, *, *, c=1, *)$。

-*AttGen*(S, upk) $\mathfrak{S}$ 根据 *upk* 在 $\mathfrak{L}$ 中找到对应的元组 $(upk, usk, hpk, hsk, *, *, *, *, c, *)$，产生 $nym, cert, ASK$，并且返回 $nym, cert, ASK$。$\mathfrak{S}$ 将该元组更新为 $(upk, usk, hpk, hsk, nym, cre, ASK, S, c, count=0)$。

-*Corrupt*(nym) 同可靠性实验。此外，$\mathfrak{S}$ 额外执行 $RL \leftarrow RL \cup (usk, nym)$。

-*Auth*(Cha, nym) 同隐私性实验。

-*Semi-Auth*(Cha, nym) $\mathfrak{S}$ 以诚实 TPM 的身份与 $\mathfrak{A}$ 联合执行 *Auth* 协议中用户端的操作。在执行过程中，$\mathfrak{A}$ 向 $\mathfrak{S}$ 发送 Host 端传送的数据，$\mathfrak{S}$ 返回 TPM 的应答。

最终，$\mathfrak{A}$ 输出 $(nym^*, \mathbb{A}^*)$。此时，$\mathfrak{S}$ 与 $\mathfrak{A}$ 共同执行 *Auth* 协议且利用 $\mathbb{A}^*$ 产生底层 CP-ABE 方案挑战密文 Cha^*。假设 $\mathfrak{A}$ 在该协议中的输出为 σ^*。若该协议获得成功，则 $\mathfrak{S}$ 检查 σ^* 是否并非借助 *Auth* 询问或 *Semi-Auth* 询问产生的。若是，则 $\mathfrak{S}$ 判定 $\mathfrak{A}$ 在实验中获胜。

模式 2

初始化 $\mathfrak{S}$ 运行 *TSetup* 算法，产生 *TPK*。$\mathfrak{S}$ 初始化列表 $\mathfrak{L}$，向 $\mathfrak{A}$ 发送 *TPK*，并且接收后者产生的 *APK*。

询问 允许 $\mathfrak{A}$ 向 $\mathfrak{S}$ 提出以下类型的预言询问。

-*USetup*() 同隐私性实验。

-*AttGen*(S, upk) 同隐私性实验。

-*Corrupt*(nym) 同模式 1。

-*Auth*(Cha, nym) 同隐私性实验。

-*Semi-Auth*(Cha, nym) 同模式 1。

最终，$\mathfrak{A}$ 输出 Cha^*, nym^*。此时，$\mathfrak{S}$ 与 $\mathfrak{A}$ 共同执行 *Auth* 协议且假设 $\mathfrak{A}$ 在该协议中的输出为 σ^*。若该协议获得成功，则 $\mathfrak{S}$ 检查是否同时满足以下条件：1）σ^* 并非借助 *Auth* 询问或 *Semi-Auth* 询问产生的；2）用于产生 σ^* 的 usk^* 是借助 *USetup* 询问产生的。若是，则判定 $\mathfrak{A}$ 在实验中获胜。

4　新的 k 次属性认证方案

4.1　方案的设计思想

本文方案可视为对 DAA 方案进行扩展得到的。*ASetup* 算法相当于 DAA 方案的系统建立算法。在该算法中，*AA* 共产生 3 个密钥对。其中，(w_1, γ_1) 是底层 CL-SDH（Camenisch-Lysyanskaya SDH）签名方案[6]的公私钥对，(w_2, γ_2) 是底层区间证明协议[20]的公私钥对。同时，$(g^{\alpha}, e(g,g)^{\beta}, \{PK_j\}_{j\in U})$, $(\alpha, g^{\beta}, \{v_j\}_{j\in U})$ 分别构成底层 CP-ABE 方案[19]的公钥和主密钥，其中，U 表示属性全集。*AttGen* 协议相当于 DAA 方案的成员注册协议。在该协议中，TPM 芯片产生两个密钥对 $(upk, usk) = (h_1^f, f), (hpk, hsk) = (h_2^y, y)$，其中，$usk$ 用于充当 TPM 私钥，且 hsk 用于产生一次性认证令牌。同时，TPM 产生知识签名 π_1，使 *AA* 确信 upk, hpk 满足正确性。最终，*User* 获得 *AA* 产生的伪名 nym，关于 (usk, hsk) 的成员证书 cre 以及关于属性集合 S_i 的底层 CP-ABE 方案解密私钥 ASK。此外，*AA* 向 *User* 提供知识签名 π_2，目的是证明所提供的 cre, ASK 满足正确性。

Auth 协议相当于 DAA 方案的签名/验证协议。在该协议中，TPM 与 Host 联合产生知识签名 π_3，从而向 *SP* 证明以下断言同时成立。1) *User* 使用的认证令牌 J_k 是利用 hsk 和公开区间 $\Phi = [1, n]$ 中尚未使用过的下标 k 产生的。2) *User* 掌握关于 (usk, hsk) 的成员证书。3) *User* 并未因 TPM 私钥 usk 泄露而被废除。4) *User* 的当前认证次数尚未超过上界 n。在断言 2) 的证明过程中，*User* 需要证明掌握成员证书 (A_i, x_i)，使 $A_i = (gh_1^f h_2^y)^{\frac{1}{\gamma_1 + x_i}}$。为了实现该项证明且不要求 *User* 执行对运算，本文借鉴了文献[20]的技术。具体地，*User* 将 A_i 盲化为 $\tilde{A}_i = A_i^{\delta}$ 的形式，并且证明掌握秘密元组 $(x_i, \delta, \delta f, \delta y)$，使得 $E_1 = \tilde{A}_i^{-x_i} g^{\delta} h_1^{\delta f} h_2^{\delta y}$。此外，该项技术要求 *SP* 额外验证是否满足 $\hat{e}(E_1, g) = \hat{e}(\tilde{A}_i, w_1)$。为了证明断言 4)，*User* 需要在 Φ 中选取未曾使用的下标 k，并且证明自己掌握关于 k 的 Boneh-Boyen 方案[22]签名 o_k。为了避免 *User* 执行对运算，本文直接采用了第 2.2 节所述的区间证明技术。通过引入预计算技术[16]，π_3 产生过程中的绝大多数运算均能在离线计算阶段完成。

为了实现对 *User* 的属性认证，本文方案借鉴了 Yang 等[10]在认证过程中使用带密钥的散列函数 $H_K()$ 的思想。具体地，*SP* 自行定义 LSSS 访问结构 $(\boldsymbol{M}, \rho)$ 并在该结构下产生关于元素 K_{SP} 的底层 CP-ABE 方案[19]挑战密文 Cha。同时，知识签名 π_3 中的挑战串 c 并非 Host 随机选取的，而是要求 Host 首先解密 Cha 得到 K_{SP}，然后由 TPM 利用 $H_K()$ 产生 c，且该过程要求将 K_{SP} 作为 $H_K()$ 的密钥。显然，若 *User* 的属性集合 S_i 无法满足 $(\boldsymbol{M}, \rho)$，所产生的知识签名 π_3 将无法通过 *SP* 的验证过程。

此外，本文方案继承了底层 CP-ABE 方案的属性更新机制，即当 *User* 的某个属性值由 j 更新为 w 时，*AA* 将为其发送更新钥 $UK_{j\to w}$。对于其他同样拥有该属性且并未执行属性更新的用户，*AA* 同样为他们发送更新钥 UK'_j。

4.2　符号定义

在本文方案的描述中，本文定义使用了多个符号。具体含义如表 1 所示。

4.3　方案的具体构造

4.3.1　系统参数产生（*TSetup*）

该算法执行以下步骤。

1) 以安全性参数 λ 作为输入，产生双线性群参数 $(G_0, G_T, p, \hat{e})$，其中 $\hat{e}$ 表示双线性映射，满足 $\hat{e}: G_0 \times G_0 \to G_T$。

2) 选取生成元 $g, h, h_1, h_2 \in_R G_0$，同时选取 $\tilde{g} \in_R G_T$。

3) 定义抗碰撞的散列函数 $H_0: \{0,1\}^* \to G_0$, $H_1: \{0,1\}^* \to \mathbb{Z}_p^*, H_2: \{0,1\}^* \to \mathbb{Z}_p^*, H_3: \{0,1\}^* \to \mathbb{Z}_p^*$,

定义带密钥的散列函数 $H_K: \{0,1\}^* \to \mathbb{Z}_p$ [10]。

4) 定义群参数 $TPK = (G_0, G_T, p, \hat{e}, g, h, h_1, h_2, \tilde{g}, H_0, H_1, H_2, H_3, H_K)$。

4.3.2　属性权威系统建立（*ASetup*）

为了产生系统公钥和主密钥，*AA* 执行以下步骤。

表 1 **本文方案使用的符号**

符号	含义	符号	含义
TPK	方案的群参数	$cert_i$	$User_i$ 的成员证书
MSK	AA 的主密钥	S_i	$User_i$ 的属性集合
APK	方案的属性公钥	ASK_i	$User_i$ 的属性私钥
Φ,Σ	底层区间证明协议中的公开列表	σ	$User_i$ 在认证阶段产生的 DAA 签名
RL	用户废除列表	(B,K)	σ 中用于对 TPM 进行废除检测的数对
$DAASeed$	TPM 的秘密种子	H_0,H_1,H_2,H_3	抗碰撞的散列函数，分别用于产生 APK，产生 TPM 私钥，产生 nym_i 以及产生知识签名 π_1,π_2,π_4 中的挑战串
cnt	TPM 的内部计数器	H_K	带密钥的散列函数，用于产生知识签名 π_3 中的挑战串
K_{AA}	AA 的身份信息	K_{SP}	SP 在认证阶段产生的挑战元素，它将充当散列函数 H_K 的密钥
nym_i	$User_i$ 的伪名	Cha	SP 产生的关于 K_{SP} 的底层 CP-ABE 方案挑战密文

1) 定义属性全集 $U=\{1,\cdots,\mathfrak{M}\}\subset\mathbb{Z}_p$。

2) 选取 $\alpha,\beta,\gamma_1,\gamma_2\in_{\mathrm{R}}\mathbb{Z}_p$。对于 $j=1,\cdots,|U|$，选取 $v_j\in_{\mathrm{R}}\mathbb{Z}_p$。设置 $w_1=g^{\gamma_1},w_2=g^{\gamma_2}$。对于 $j=1,\cdots,|U|$，设置 $PK_j=H_0(j)^{v_j}$。设置 $MSK=(\alpha,g^{\beta},\{v_j\}_{j\in U},\gamma_1,\gamma_2)$。

3) 定义集合 $\Phi=\{1,2,\cdots,n\},\Sigma=\{o_1,o_2,\cdots,o_n\}=\{g^{\frac{1}{\gamma_2+1}},g^{\frac{1}{\gamma_2+2}},\cdots,g^{\frac{1}{\gamma_2+n}}\}$。

4) 初始化用户废除列表 $RL\leftarrow\varnothing$。

5) 设置 $APK=(w_1,w_2,g^{\alpha},\hat{e}(g,g)^{\beta},\{PK_j\}_{j\in U},\Phi,\Sigma,RL)$。

4.3.3 *属性密钥产生（AttGen）*

若 $User_i$(Host + TPM) 希望获得关于属性集合 S_i 的底层 CP-ABE 方案属性私钥，他需要与 AA 共同执行以下步骤。同时，假设 TPM 已经利用签署密钥与 AA 建立了安全的认证信道。

1) Host 向 AA 发送请求的属性集合 S_i，后者返回随机消息 $n_{AA}\in_{\mathrm{R}}\{0,1\}^{\iota}$。

2) TPM 设置 $cnt=cnt+1$，计算 $f=H_1(DAASeed\|cnt\|K_{AA}),F=h_1^f$。TPM 选取 $y\in_{\mathrm{R}}\mathbb{Z}_p$，计算 $Y=h_2^y$。TPM 产生知识签名 $\pi_1=SPK\{(f,y):F=h_1^f\wedge Y=h_2^y\}(n_{AA})$，并且通过 Host 向 AA 发送 F,Y,π_1。需要指出的是，步骤 1)和步骤 2)可视为用户端的密钥产生算法 $USetup$。

3) 对于每个 $f'\in RL$，AA 检查是否满足 $F=h_1^{f'}$。若是，则 AA 终止协议。否则，AA 进一步验证 π_1 的有效性。

4) 若 π_1 有效，则 AA 选取 $x_i\in_{\mathrm{R}}\mathbb{Z}_p$，设置伪名 $nym_i=H_2(F)$，计算 $A_i=(gFY)^{\frac{1}{\gamma_1+x_i}},D=g^{\beta+\alpha x_i}$，$D'=g^{(\alpha+\beta+\alpha x_i)nym_i}$。同时，对于每个属性 $j\in S_i$，AA 选取 $r_j\in_{\mathrm{R}}\mathbb{Z}_p$，计算 $D_j=g^{x_i}PK_j^{r_j}$，$D_j'=(g^{\alpha})^{r_j}$。

5) AA 向 TPM 发送 $nym_i,cert_i=(A_i,x_i)$，$ASK_i=(S_i,D,D',\{D_j,D_j'\}_{j\in S_i})$ 以及知识签名 π_2，即

$$\pi_2=SPK\{(\gamma_1,\alpha,\{r_j\}_{j\in S_i}):A_i^{-x_i}gFY=A_i^{\gamma_1}\wedge w_1=g^{\gamma_1}\wedge\hat{e}(D,g)\cdot\hat{e}(g,g)^{-\beta}=\hat{e}(g^{x_i},g)^{\alpha}\wedge\{D_j\cdot g^{-x_i}=PK_j^{r_j}\}_{j\in S_i}\wedge\{D_j'=(g^{\alpha})^{r_j}\}_{j\in S_i}\}$$

6) TPM 向 Host 发送 $(Y,y,nym_i,cert_i,ASK_i,\pi_2)$，后者验证 π_2 的有效性以及是否满足 $\hat{e}(D,g)\ \hat{e}(g^{\alpha},g)=\hat{e}(D',g)^{nym_i^{-1}}$。若是，则保存 $(Y,y),nym_i,cert_i,ASK_i$。

$User_i$ 与 AA 在 $AttGen$ 协议中的主要交互过程如图 1 所示。

4.3.4 *属性认证（Auth）*

用户 $User_i$(TPM+Host) 与 SP 执行以下交互过程。

1) SP 选取 $s,s^*\in_{\mathrm{R}}\mathbb{Z}_p$，选取 $K_{SP}\in_{\mathrm{R}}G_T$，计算 $\tilde{C}=K_{SP}\ \hat{e}(g,g)^{\beta s},C=g^s$。

2) SP 定义 LSSS 访问结构 $(\boldsymbol{M},\rho)$，其中 $\boldsymbol{M}$ 为 $l\times\bar{n}$ 矩阵，ρ 为映射，即将 $\boldsymbol{M}$ 的第 i 行 $\boldsymbol{M}_i$ 映射为属性 $\rho(i)$。SP 选取 $y_2,y_3,\cdots,y_n,y_2',y_3',\cdots,y_n'\in_{\mathrm{R}}\mathbb{Z}_p$，定义随机列向量 $\vec{\boldsymbol{v}}=(s,y_2,y_3,\cdots,y_n)$，$\vec{\boldsymbol{v}}'=(s^*,y_2',y_3',\cdots,y_n')$。

3) 对于 $i=1,\cdots,l$，SP 计算 $C_i=(g^{\alpha})^{\boldsymbol{M}_i\vec{\boldsymbol{v}}}=g^{\alpha\boldsymbol{M}_i\vec{\boldsymbol{v}}}$，$C_i'=PK_{\rho(i)}^{\boldsymbol{M}_i\vec{\boldsymbol{v}}}=H_0(\rho(i))^{v_{\rho(i)}\boldsymbol{M}_i\vec{\boldsymbol{v}}},C_i''=g^{\boldsymbol{M}_i\vec{\boldsymbol{v}}'}$。

4) 假设当前 RL 的内容为 $RL=\{(f_1,nym_1^*),(f_2,nym_2^*),\cdots\}$。对于 $\iota=1,\cdots,|RL|$，SP 选取 $s_{\iota}\in\mathbb{Z}_p$，

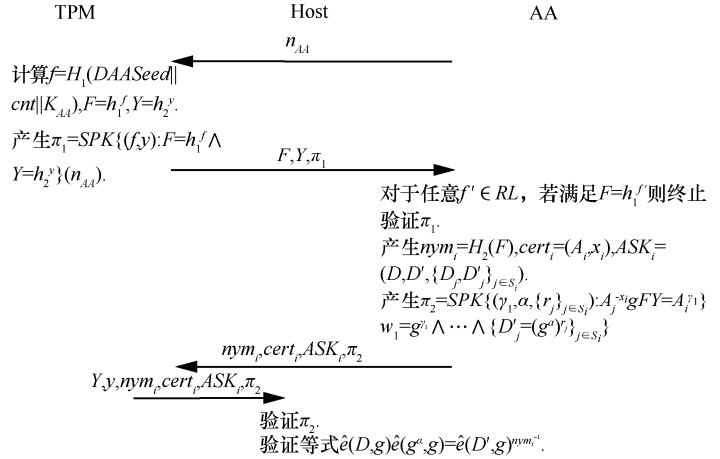

图 1　*AttGen* 协议的交互过程

使 $\sum_{t=1}^{|RL|} s_t = s^*$。对于 $t=1,\cdots,|RL|$，*SP* 设置 $C^*_{1,t} = g^{-s_t \cdot nym_t^*}, C^*_{2,t} = g^{s_t}$。

5) *SP* 选取 $n_{SP} \in_R \{0,1\}^t$，并且向 Host 发送 $Cha = ((\boldsymbol{M},\rho),\tilde{C},C,\{C_i,C'_i,C''_i\}_{i=1}^l,\{C^*_{1,t},C^*_{2,t}\}_{t=1}^{|RL|}),n_{SP}$。

6) Host 将 *Cha* 分离为 $Cha = ((\boldsymbol{M},\rho),\tilde{C},C,\{C_i,C'_i,C''_i\}_{i=1}^l,\{C^*_{1,t},C^*_{2,t}\}_{t=1}^{|RL|})$ 的形式。若 $S_i \mid \neq (\boldsymbol{M},\rho)$，则 Host 输出 $\perp$，并终止当前协议。

7) 令 $I=\{i:\rho(i)\in S_i\}$。Host 计算常量集合 $\{\omega_i \in \mathbb{Z}_p\}_{i\in I}$，使得 $\sum_{i\in I}\omega_i \boldsymbol{M}_i = (1,0,0,\cdots,0)$。Host 设置 $D''=Dg^\alpha$，计算

$$K_{SP} = \frac{\tilde{C}\prod_{i\in I}\left(\frac{\hat{e}(D_{\rho(i)},C_i^{\omega_i})\hat{e}(D'',C_i''^{\omega_i})}{\hat{e}(D'_{\rho(i)},C_i'^{\omega_i})}\right)}{\hat{e}(D,C)\prod_{t=1}^{|RL|}\left(\hat{e}(D'',C^*_{1,t})\,\hat{e}(D',C^*_{2,t})\right)^{\frac{1}{nym-nym_t^*}}}$$

8) Host 选取新鲜下标 $k \in_R \boldsymbol{\Phi}$，计算认证令牌 $J_k = \tilde{g}^{\frac{1}{y+k+1}}$。

9) Host 选取 $\delta \in_R \mathbb{Z}_p$，计算 $\tilde{A}_i = A_i^\delta$，$E_1 = \tilde{A}_i^{-x_i}(gFY)^\delta$。Host 选取 $l,\nu \in_R \mathbb{Z}_p$，计算 $\tilde{o}_k = o_k^l$，$E_2 = \tilde{o}_k^{-k}g^l, E_3 = g^k h^\nu$，并且与 TPM 联合产生以下形式的知识签名

$$\pi_3 = SPK\{(y,k,x_i,\delta,\delta f,\delta y,f,l,\nu): J_k\tilde{g}^{-1} = J_k^{-y}J_k^{-k} \wedge E_1 = \tilde{A}_i^{-x_i}g^\delta h_1^{\delta f}h_2^{\delta y} \wedge K = B^f \wedge E_2 = \tilde{o}_k^{-k}g^l \wedge E_3 = g^k h^\nu\}(n_{SP})$$

然后，Host 向 *SP* 发送 $\sigma = (\pi_3,J_k,\tilde{A}_i,B,K,\tilde{o}_k,E_1,E_2,E_3)$。

10) *SP* 将 σ 分离为 $\sigma = (\pi_3,J_k,\tilde{A}_i,B,K,\tilde{o}_k,E_1,E_2,E_3)$ 的形式，并且执行以下的验证过程。①对于每个 $f' \in RL$，验证是否满足 $K \neq B^{f'}$。②验证 π_3 的有效性。③验证是否满足 $\hat{e}(E_1,g)=\hat{e}(\tilde{A}_i,w_1)$, $\hat{e}(E_2,g)=\hat{e}(\tilde{o}_k,w_2)$。④验证 $\tilde{o}_k$ 是否并非群 G_0 的单位元。若上述验证都通过，则接受用户的认证过程。

$User_i$ 与 *SP* 在 *Auth* 协议中的主要交互过程如图 2 所示。

4.3.5　属性更新（*AUpdate*）

假设 $User_i$(TPM+Host) 希望将属性 j 更新为 w。为此，他需要与 *AA* 执行以下交互过程。

1) Host 向 *AA* 发送更新请求 $req_{j\to w}$，后者返回随机消息 $n_{AA} \in_R \{0,1\}^t$。

2) Host 选取 $\delta \in_R \mathbb{Z}_p$，计算 $\tilde{A}_i = A_i^\delta, E_1 = \tilde{A}_i^{-x_i}(gFY)^\delta$。然后，Host 与 TPM 联合产生知识签名 $\pi_4 = SPK\{(x_i,\delta,\delta f,\delta y,f): E_1 = \tilde{A}_i^{-x_i}g^\delta h_1^{\delta f}h_2^{\delta y} \wedge K = B^f\}(n_{AA})$，并向 *AA* 发送 $\pi_4,\tilde{A}_i,B,K,E_1$。

3) 对于每个 $f' \in RL$，*AA* 检查是否满足 $K = B^{f'}$。若是，则 *AA* 终止协议。否则，*AA* 检查 π_4 的有效性以及是否满足 $\hat{e}(E_1,g)=\hat{e}(\tilde{A}_i,w_1)$。

4) 若上述检查通过，*AA* 选取 $r_w,\tilde{v}_j \in_R \mathbb{Z}_p$，向 Host 返回 $UK_{j\to w} = (UK'_{j\to w},UK''_{j\to w}) = (H_0(j)^{-r_j v_j} \cdot H_0(w)^{r_w v_w}, \frac{r_w}{r_j})$。同时，*AA* 向其他同样拥有属性 j 且并未执行属性更新的用户 $User_i'$ 发送 $UK'_j = H_0(j)^{r_j(\tilde{v}_j - v_j)}$，并将 *APK* 中的元素 $PK_j = H_0(j)^{v_j}$ 更新为 $PK_j = H_0(j)^{\tilde{v}_j}$。

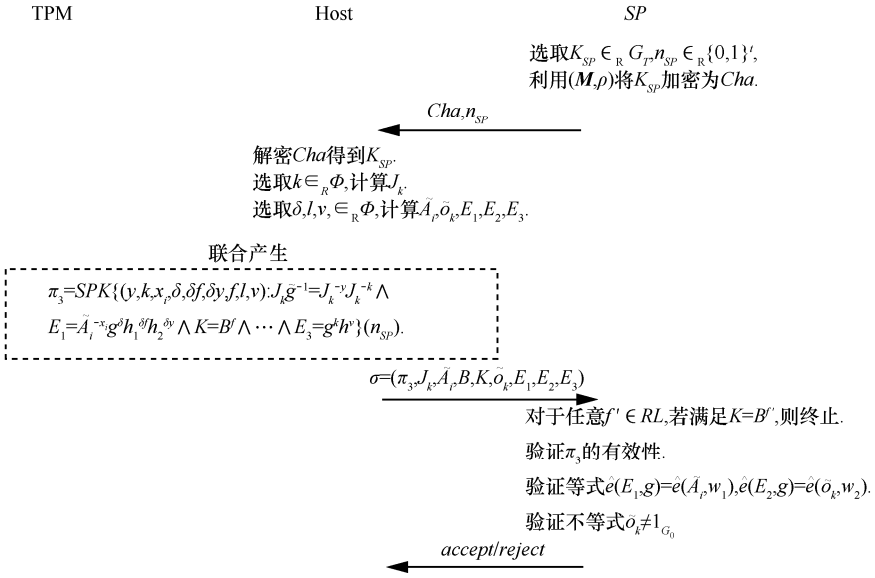

图 2　*Auth* 协议的交互过程

5) 在接收到$UK_{j\to w}$之后，Host 将 ASK_i 中的元素 $D_j=g^{x_i}H_0(j)^{r_jv_j}, D'_j=(g^{\alpha})^{r_j}$ 更新为 $D_w=D_j\cdot UK'_{j\to w}=g^{x_i}H_0(w)^{r_wv_w}, D'_w=(D'_j)^{UK''_{j\to w}}=(g^{\alpha})^{r_w}$。对于其他用户 $User'_i$，则可以利用 UK'_j 将其 ASK'_i 中的对应元素更新为 $\tilde{D}_j=g^{x'_i}H_0(j)^{v_jr_j}UK'_j=g^{x'_i}H_0(j)^{r_j\tilde{v}_j}$, $\tilde{D}'_j=D'_j=(g^{\alpha})^{r_j}$。

4.3.6　用户废除（*Revoke*）

假设 $User_i$ 被攻破且 $usk_i=f, hsk_i=y, cert_i=(A_i,x_i)$ 均遭泄露。此时，*AA* 执行以下步骤。

1) 验证是否满足 $\hat{e}(A_i,w_1g^{x_i})=\hat{e}(gh_1^fh_2^y,g)$。

2) 若是，则计算 $nym=H_2(h_1^f)$，并且设置 $RL\leftarrow RL\cup(f,nym)$。

5　底层知识签名的详细描述

在上一节的方案描述中，共涉及 4 个知识签名。其中，π_1 与 π_2 分别由 TPM 和 *AA* 采用标准方式产生，π_3,π_4 是由 TPM 与 Host 联合产生的。限于篇幅，本节仅详细介绍 π_3 的产生和验证过程。需要指出的是，在 π_1 的产生过程中，TPM 向 Host 提供数对 $(h_1,F=h_1^f)$。在 π_3,π_4 的产生过程中，TPM 向 Host 提供数对 $(B,K=B^f)$。由于 h_1 是群参数中的公开元素且 B 是由 TPM 随机选取的元素。因此，Host 无法对 h_1 与 B 的取值进行控制，从而有效地去除了静态 Diffie-Hellman 预言机[17-18]。此外，为了进一步优化运算效率，本文采用了文献[16]中的预计算技术，即 π_3 产生过程的步骤 1)~步骤 3)可以预先以离线方式执行，从而减轻了用户端的在线运算负担。

π_3 的具体产生步骤如下所示。

1) Host 选取 $\rho_y,\rho_k,\rho_{x_i},\rho_\delta,\rho_{\delta y},\rho_l,\rho_v\in_R\mathbb{Z}_p$，并且向 TPM 发送 $\rho_y,\rho_k,\rho_{x_i},\rho_\delta,\rho_{\delta y},\rho_l,\rho_v$，$y,k,x_i,\delta,l,v,J_k,\tilde{A}_i,\tilde{o}_k,E_1,E_2,E_3$。

2) TPM 选取 $B\in_R G_T$，计算 $K=B^f$，其中，$f=H_1(DAASeed\|cnt\|K_{AA})$。此外，TPM 选取 $\rho_{\delta f},\rho_f\in_R\mathbb{Z}_p$，计算 $R_{2t}=h_1^{\rho_{\delta f}}, R_3=B^{\rho_f}$，并且向 Host 返回 R_{2t},B,K。

3) Host 计算 $R_1=J_k^{-\rho_y}J_k^{-\rho_k}, R_2=\tilde{A}_i^{-\rho_{x_i}}g^{\rho_\delta}R_{2t}h_2^{\rho_{\delta y}}$, $R_4=\tilde{o}_k^{-\rho_k}g^{\rho_l}, R_5=g^{\rho_k}h^{\rho_v}$。

4) Host 向 TPM 发送 $R_1,R_2,R_4,R_5,n_{SP},K_{SP}$。

5) TPM 设置 $c=H_{K_{SP}}(J_k\|\tilde{A}_i\|B\|K\|\tilde{o}_k\|E_1\|E_2\|E_3\|R_1\|R_2\|\cdots\|R_5\|n_{SP})$，在 $\mathbb{Z}_p$ 上计算 $s_y=\rho_y+cy$, $s_k=\rho_k+ck, s_{x_i}=\rho_{x_i}+cx_i, s_\delta=\rho_\delta+c\delta, s_{\delta f}=\rho_{\delta f}+c\delta f, s_{\delta y}=\rho_{\delta y}+c\delta y, s_f=\rho_f+cf, s_l=\rho_l+cl$, $s_v=\rho_v+cv$，并且输出 $\pi_3=(c,s_y,s_k,s_{x_i},s_\delta,s_{\delta f},s_{\delta y},s_f,s_l,s_v)$。

π_3 的具体验证步骤如下所示。

1) *SP* 将 π_3 分离为 $\pi_3=(c,s_y,s_k,s_{x_i},s_\delta,s_{\delta f},s_{\delta y},s_f,s_l,s_v)$ 的形式。

2) *SP* 计算 $\hat{R}_1=J_k^{-s_y}J_k^{-s_k}(J_k\tilde{g}^{-1})^{-c}$, $\hat{R}_2=\tilde{A}_i^{-s_{x_i}}\cdot g^{s_\delta}h_1^{s_{\delta f}}h_2^{s_{\delta y}}E_1^{-c}, \hat{R}_3=B^{s_f}K^{-c}$, $\hat{R}_4=\tilde{o}_k^{-s_k}g^{s_l}E_2^{-c}$, $\hat{R}_5=$

$g^{s_k}h^{s_v}E_3^{-c}$。

3) *SP* 验证是否满足 $c=H_{K_{SP}}(J_k\|\tilde{A}_i\|B\|K\|\tilde{o}_k\|E_1\|E_2\|E_3\|\hat{R}_1\|\hat{R}_2\|\cdots\|\hat{R}_5\|n_{SP})$。若是，则接受；否则，拒绝。

6 支持外包解密运算的改进方案

在改进方案中，为了将底层 CP-ABE 方案[19]解密过程中的主要运算外包给云服务器 *Server*，需要对第 4 节的 *TSetup* 算法和 *Auth* 协议进行修改。同时，增加算法 $AttGen_{\text{out}}$, $Transform_{\text{out}}$, $Decrypt_{\text{out}}$。其中，$AttGen_{\text{out}}$ 算法的作用是允许 *User* 利用属性私钥 *ASK* 自行产生转换钥 *TK*。$Transform_{\text{out}}$ 算法的作用是允许 *Server* 利用 *TK* 对 *Cha* 执行解密，得到部分解密的密文 Cha'。$Decrypt_{\text{out}}$ 算法的作用是允许 *User* 利用 *ASK* 将 Cha' 转换为最终的明文 K_{SP}。

6.1 系统参数产生（*TSetup*）

在第 4 节 *TSetup* 算法的基础上，需要补充定义以下的散列函数，即 $H_4:\{0,1\}^*\to\mathbb{Z}_p^*, H_5:\{0,1\}^*\to G_T, H_6:\{0,1\}^*\to\mathbb{Z}_p^*$，即将 *TPK* 扩充为 $TPK=(G_0,G_T,p,\hat{e},g,h,h_1,h_2,\tilde{g},H_0,H_1,\cdots,H_6,H_K)$ 的形式。

6.2 属性认证（*Auth*）

与第 4 节的 *Auth* 协议相比，需要做出以下修改。在步骤 1），*SP* 额外选取 $R\in_R G_T$，设置 $s=H_4(R,K_{SP})$，计算 $\tilde{C}=R\cdot\hat{e}(g,g)^{\beta s}, C=g^s$。在步骤 2）*SP* 额外设置 $\bar{r}=H_5(R)$，并且计算 $\hat{C}=K_{SP}\oplus\bar{r}, tag=H_6(\bar{r})$。在步骤 5），*SP* 向 Host 发送 $Cha=((\boldsymbol{M},\rho),\tilde{C},C,\hat{C},\{C_i,C_i',C_i''\}_{i=1}^{l},\ \{C_{1,t}^*,C_{2,t}^*\}_{t=1}^{|RL|}, tag), n_{SP}$。在步骤 7），Host 向 *Server* 发送 TK, Cha，后者调用 $Transform_{\text{out}}$ 算法并返回部分解密结果 Cha'。然后，Host 调用 $Decrypt_{\text{out}}$ 算法，从而得到 K_{SP}。

6.3 转换密钥产生（$AttGen_{\text{out}}$）

假设 $User_i$(Host + TPM) 通过执行第 4 节 *AttGen* 协议得到的属性私钥符合以下形式，即 $ASK_i'=(S_i,\tilde{D}=g^{\beta+\alpha x_i},\tilde{D}'=g^{(\alpha+\beta+\alpha x_i)nym_i},\{\tilde{D}_j=g^{x_i}H_0(j)^{v_jr_j},\tilde{D}_j'=g^{\alpha r_j}\}_{j\in S_i})$。为了获得转换钥 *TK*，$User_i$ 执行以下步骤。

1) Host 选取 $z\in_R\mathbb{Z}_p^*$，并且设置 $TK=(S_i, D=\tilde{D}^{\frac{1}{z}},\ D'=\tilde{D}'^{\frac{1}{z}},\ D''=(\tilde{D}\cdot g^{\alpha})^{\frac{1}{z}},\{D_j=\tilde{D}_j^{\frac{1}{z}},D_j'=\tilde{D}_j'^{\frac{1}{z}}\}_{j\in S_i})$。

2) Host 保存 $ASK_i=(z,TK)$。

6.4 密文转换（$Transform_{out}$）

在接收到 Host 提供的 Cha, TK 之后，*Server* 执行以下步骤。

1) 将 *Cha* 分离为 $Cha=((\boldsymbol{M},\rho),\tilde{C},C,\hat{C},\{C_i,C_i',C_i''\}_{i=1}^{l},\{C_{1,t}^*,C_{2,t}^*\}_{t=1}^{|RL|},tag)$ 的形式。

2) 假设 $S_i\models(\boldsymbol{M},\rho)$，否则 *Server* 将输出 $\perp$。定义 $I=\{i:\rho(i)\in S_i\}$。计算常量集合 $\{\omega_i\in\mathbb{Z}_p\}_{i\in I}$，使 $\sum_{i\in I}\omega_i\boldsymbol{M}_i=(1,0,0,\cdots,0)$。计算

$$T_2=\frac{\hat{e}(D,C)\prod_{t=1}^{|RL|}\left(\hat{e}(D'',C_{1,t}^*)\hat{e}(D',C_{2,t}^*)\right)^{\frac{1}{nym-nym_t^*}}}{\prod_{i\in I}\left(\frac{\hat{e}(D_{\rho(i)},C_i^{\omega_i})\hat{e}(D'',C_i''^{\omega_i})}{\hat{e}(D'_{\rho(i)},C_i'^{\omega_i})}\right)}=\hat{e}(g,g)^{\frac{\beta s}{z}}$$

3) 设置 $T_0=\tilde{C}=R\hat{e}(g,g)^{\beta s}, T_1=\hat{C}=K_{SP}\oplus\bar{r}$，输出 $Cha'=(T_0,T_1,T_2)$。

6.5 解密（$Decrypt_{\text{out}}$）

当接收到 *Server* 返回的 Cha'，Host 执行以下步骤。

1) 将 *Cha* 分离为 $Cha=((\boldsymbol{M},\rho),\tilde{C},C,\ \hat{C},\{C_i,C_i',C_i''\}_{i=1}^{l},\{C_{1,t}^*,C_{2,t}^*\}_{t=1}^{|RL|},tag)$ 的形式，同时将 Cha' 分离为 $Cha'=(T_0,T_1,T_2)$ 的形式。

2) 计算 $R=\frac{T_0}{T_2^z},\bar{r}=H_5(R),K_{SP}=T_1\oplus\bar{r}, s=H_4(R,K_{SP})$。

3) 验证是否满足 $T_0=R(\hat{e}(g,g)^{\beta})^s,\ T_2=(\hat{e}(g,g)^{\beta})^{\frac{s}{z}}, tag=H_6(\bar{r})$。若是，则输出 K_{SP}，否则，输出 $\perp$。

7 安全性证明

定理 1　第 4 节方案满足正确性。

证明过程见本文附录。

定理 2　在随机预言模型下，只要群 G_0, G_T 上的 DBDH,DBDHI 和 q-SDH 假设成立，则第 4 节方案在新的 k-TABA 模型下是安全的。

证明　在本文方案的证明过程中，模拟器 $\mathfrak{S}$ 需要定义多个列表，用于对各类预言机进行模拟。具体地，$\mathfrak{L}_K$ 用于对散列预言机 $H_K(\cdot)$ 进行模拟，$\mathfrak{L}_i(i=2,3)$ 用于对散列预言机 $H_i(\cdot)$ $(i=2,3)$ 进行模拟，$\mathfrak{L}$ 用于对预言机 $USetup(), AttGen(\cdot,\cdot), Corrupt(\cdot)$ 进行模拟。$\mathfrak{L}_{Auth}$ 用于对预言机 $Auth(\cdot,\cdot), Semi\text{-}Auth(\cdot,\cdot)$ 进行模拟。$\mathfrak{L}_{nym}$ 用于对预言机 $AttGen(\cdot,\cdot)$ 进行模拟。

隐私性 当前实验的执行过程分以下两种模式。

模式 1 $\mathfrak{S}$ 以 DBDH 问题实例 $(\overline{g},\overline{g}^{a},\ \overline{g}^{b},\ \overline{g}^{\varsigma},\overline{Z}=\hat{e}(\overline{g},\overline{g})^{abc\varsigma},\ \overline{R}=\hat{e}(\overline{g},\overline{g})^{d})\in G_0^4\times G_T^2$ 作为输入，其目标是对以下情况进行分辨：1）满足 $\overline{Z}=\hat{e}(\overline{g},\overline{g})^{abc\varsigma}$；2）满足 $\overline{R}=\hat{e}(\overline{g},\overline{g})^{abc\varsigma}$。其中，$a,b,\varsigma,d\in\mathbb{Z}_p^*$。在初始化阶段，$\mathfrak{S}$ 设置 $h_1=\overline{g}$，并且采用 *TSetup* 算法中的方式产生 *TPK* 中的剩余元素。$\mathfrak{S}$ 执行 *ASetup* 算法，产生 MSK,APK。同时，$\mathfrak{S}$ 选取 $\alpha,\beta\in_{\mathrm{R}}\{1,\cdots,m\}$，其中，$m$ 表示系统中的最大用户数量。$\mathfrak{S}$ 初始化列表为 $\mathfrak{L},\mathfrak{L}_K,\mathfrak{L}_2,\mathfrak{L}_3$，并且向 $\mathfrak{A}$ 发送 TPK,MSK,APK。在询问 1 阶段，$\mathfrak{S}$ 采用以下方式对各类预言机进行模拟。

-$H_K(m)$ 若 $(m,h_K)\in\mathfrak{L}_K$，则 $\mathfrak{S}$ 返回 h_K。否则，$\mathfrak{S}$ 选取 $h_K\in_{\mathrm{R}}\mathbb{Z}_p^*$，并且返回 h_K。同时，$\mathfrak{S}$ 执行 $\mathfrak{L}_K\leftarrow\mathfrak{L}_K\cup(m,h_K)$。

-$H_2(m)$ 若 $(m,h_2)\in\mathfrak{L}_2$，则 $\mathfrak{S}$ 返回 h_2。否则，$\mathfrak{S}$ 选取 $h_2\in_{\mathrm{R}}\mathbb{Z}_p^*$，并且返回 h_2。同时，$\mathfrak{S}$ 执行 $\mathfrak{L}_2\leftarrow\mathfrak{L}_2\cup(m,h_2)$。

-$H_3(m)$. 若 $(m,h_3)\in\mathfrak{L}_3$，则 $\mathfrak{S}$ 返回 h_3。否则，$\mathfrak{S}$ 选取 $h_3\in_{\mathrm{R}}\mathbb{Z}_p^*$，并且返回 h_3。同时，$\mathfrak{S}$ 执行 $\mathfrak{L}_3\leftarrow\mathfrak{L}_3\cup(m,h_3)$。

-USetup() 对该询问的模拟分以下 3 种情况。

情况 1 若当前为第 α 次询问，则 $\mathfrak{S}$ 选取 $u_\alpha\in_{\mathrm{R}}\mathbb{Z}_p^*$，设置 $upk_\alpha=F_\alpha=(\overline{g}^{a})^{u_\alpha}$。$\mathfrak{S}$ 选取 $y_\alpha\in_{\mathrm{R}}\mathbb{Z}_p^*$，设置 $hpk_\alpha=Y_\alpha=h_2^{y_\alpha}$，并且返回 upk_α,hpk_α。同时，$\mathfrak{S}$ 执行 $\mathfrak{L}\leftarrow\mathfrak{L}\cup(upk_\alpha,usk_\alpha,hpk_\alpha,hsk_\alpha,*,*,*,*,*,*)$。显然，$usk_\alpha=au_\alpha$ 为未知元素。

情况 2 若当前为第 β 次询问，则 $\mathfrak{S}$ 选取 $u_\beta\in_{\mathrm{R}}\mathbb{Z}_p^*$，设置 $upk_\beta=F_\beta=(\overline{g}^{a})^{u_\beta}$。$\mathfrak{S}$ 选取 $y_\beta\in_{\mathrm{R}}\mathbb{Z}_p^*$，设置 $hpk_\beta=Y_\beta=h_2^{y_\beta}$，并且返回 upk_β,hpk_β。同时，$\mathfrak{S}$ 执行 $\mathfrak{L}\leftarrow\mathfrak{L}\cup(upk_\beta,usk_\beta,\ hpk_\beta,hsk_\beta,*,*,*,*,*,*)$。显然，$usk_\beta=au_\beta$ 为未知元素。

情况 3 在其他情况下，$\mathfrak{S}$ 选取 $f,y\in_{\mathrm{R}}\mathbb{Z}_p^*$，设置 $upk=F=h_1^{f},hpk=Y=h_2^{y}$，并且返回 upk,hpk。同时，$\mathfrak{S}$ 执行 $\mathfrak{L}\leftarrow\mathfrak{L}\cup(upk,usk,hpk,\ hsk,*,*,*,*,*,*)$。

-AttGen(S,upk). 对该询问的模拟分以下 3 种情况。

情况 1 若 $upk=upk_\alpha$，则 $\mathfrak{S}$ 采用模拟方式产生知识签名 π_1。在 *AttGen* 协议结束后，$\mathfrak{S}$ 获得由 $\mathfrak{A}$ 产生的 $nym_\alpha,cre_\alpha,ASK_\alpha$。$\mathfrak{S}$ 将用户 upk_α 在 $\mathfrak{L}$ 中的对应表目更新为 $(upk_\alpha,usk_\alpha,hpk_\alpha,\ hsk_\alpha,\ nym_\alpha,\ cre_\alpha,\ ASK_\alpha,S,c_\alpha=0,count=0)$。

情况 2 若 $upk=upk_\beta$，则 $\mathfrak{S}$ 采用模拟方式产生知识签名 π_1。在 *AttGen* 协议结束后，$\mathfrak{S}$ 获得由 $\mathfrak{A}$ 产生的 $nym_\beta,cre_\beta,ASK_\beta$。$\mathfrak{S}$ 将用户 upk_β 在 $\mathfrak{L}$ 中的对应表目更新为 $(upk_\beta,usk_\beta,hpk_\beta,\ hsk_\beta,nym_\beta,\ cre_\beta,ASK_\beta,S,c_\beta=0,count=0)$。

情况 3 若 $upk\notin\{upk_\alpha,upk_\beta\}$，则 $\mathfrak{S}$ 根据 upk 在 $\mathfrak{L}$ 中找到对应的表目，并利用 usk,hsk 以诚实方式产生知识签名 π_1。剩余操作同情况 1 和情况 2。

-Corrupt(nym) $\mathfrak{S}$ 根据 nym 在 $\mathfrak{L}$ 中找到对应的表目 $(upk,usk,hpk,hsk,nym,cre,ASK,S,c,\ count)$。若 $upk\in\{upk_\alpha,upk_\beta\}$，则 $\mathfrak{S}$ 模拟失败。否则，若 $c=0$，$\mathfrak{S}$ 返回 usk,hsk，并且将该表目更新为 $(upk,usk,hpk,hsk,nym,cre,ASK,S,c=1,count)$。

-Auth(Cha,nym) $\mathfrak{S}$ 根据 nym 在 $\mathfrak{L}$ 中找到表目 $(upk,usk,hpk,hsk,nym,cre,ASK,S,c,count)$。同时，$\mathfrak{S}$ 从密文 *Cha* 中分离出访问结构 $(\boldsymbol{M},\rho)$，并且检查是否满足 $S\models(\boldsymbol{M},\rho)$ 且 $count<n$。若是，则 $\mathfrak{S}$ 利用 *ASK* 从密文 *Cha* 中恢复出秘密元素 K_{SP}。接下来的模拟过程分以下 3 种情况。

情况 1 若 $nym=nym_\alpha$，则 $\mathfrak{S}$ 选取 $v\in_{\mathrm{R}}\mathbb{Z}_p^*$，设置 $B=\hat{e}(\overline{g}^{b},\overline{g}^{\varsigma})^{v},K=\overline{Z}^{u_\alpha v}$。$\mathfrak{S}$ 选取 $\tilde{A}_\alpha,\ \tilde{o}_k\in_{\mathrm{R}}G_0$，从底层知识证明 $PK\{(\gamma_1^*,\gamma_2^*):\hat{e}(\tilde{A}_\alpha,w_1)=\hat{e}(\tilde{A}_\alpha,g)^{\gamma_1^*}\wedge\hat{e}(\tilde{o}_k,w_2)=\hat{e}(\tilde{o}_k,g)^{\gamma_2^*}\}$ 中提取出知识 γ_1^*,γ_2^*，并且设置 $E_1=\tilde{A}_\alpha^{\gamma_1^*},\ E_2=\tilde{o}_k^{\gamma_2^*}$。$\mathfrak{S}$ 选取 $E_3\in_{\mathrm{R}}G_0,J_k\in_{\mathrm{R}}G_T$，选取 $c,s_y,s_k,s_{x_\alpha},s_\delta,s_{\delta f_\alpha},s_{\delta y_\alpha},\ s_{f_\alpha},s_l,s_v\in_{\mathrm{R}}\mathbb{Z}_p$，并且采用知识签名 π_3 验证过程中的方式计算 $\hat{R}_1,\ \hat{R}_2,\cdots,\ \hat{R}_5$。$\mathfrak{S}$ 定义 $H_{K_{SP}}(J_k\|\ \tilde{A}_\alpha\|B\|K\|\tilde{o}_k\|E_1\|E_2\|E_3\|\hat{R}_1\|\ \hat{R}_2\|\cdots\|\hat{R}_5\|n_{SP})=c$，返回 $\sigma=(\pi_3,J_k,\tilde{A}_\alpha,\ B,K,\tilde{o}_k,E_1,E_2,E_3)$。

情况 2 若 $nym=nym_\beta$，则 $\mathfrak{S}$ 采用类似的方式模拟产生 σ。2 种情况的区别是，$\mathfrak{S}$ 设置 $K=\overline{R}^{u_\beta v}$。

情况 3 若 $nym\notin\{nym_\alpha,nym_\beta\}$，则 $\mathfrak{S}$ 采用诚实方式产生并返回 σ。

无论属于哪种情况，$\mathfrak{S}$ 都需要在认证过程结束后将表目 $(upk,usk,hpk,hsk,nym,cre,ASK,\ S,c,count)$ 更新为 $(upk,usk,hpk,hsk,nym,\ cre,\ ASK,S,c,\ count+1)$。

在询问 1 阶段的末尾，$\mathfrak{A}$ 输出 $(nym_0^*,nym_1^*,$

Cha^*）。$\mathfrak{S}$ 检查是否满足 $\{nym_0^*, nym_1^*\} = \{nym_\alpha, nym_\beta\}$。若否，则 $\mathfrak{S}$ 模拟失败。同时，$\mathfrak{S}$ 从密文 Cha^* 中分离出访问结构 $(\boldsymbol{M}^*, \rho^*)$。

接下来，$\mathfrak{S}$ 根据 nym_0^*, nym_1^* 分别从 $\mathfrak{L}$ 中找到对应的元组，并且检查是否同时满足 $S_0^* \models (\boldsymbol{M}^*, \rho^*)$, $S_1^* \models (\boldsymbol{M}^*, \rho^*), c_0^* = c_1^* = 0$ 且 $count^* < n$。若是，则 $\mathfrak{S}$ 选取 $\bar{b} \in_R \{0,1\}$。若 $\bar{b} = 0$，则 $\mathfrak{S}$ 以用户 nym_α 的身份与 $\mathfrak{A}$ 执行 *Auth* 协议。具体模拟方式同 *Auth* 询问的情况 1。若 $\bar{b} = 1$，则 $\mathfrak{S}$ 以用户 nym_β 的身份与 $\mathfrak{A}$ 执行 *Auth* 协议。具体模拟方式同 *Auth* 询问的情况 2。同时，$\mathfrak{S}$ 将 $\mathfrak{L}$ 中的表目 $(upk_{\bar{b}}^*, usk_{\bar{b}}^*, hpk_{\bar{b}}^*, hsk_{\bar{b}}^*, nym_{\bar{b}}^*, cre_{\bar{b}}^*, ASK_{\bar{b}}^*, S_{\bar{b}}^*, c_{\bar{b}}^*, count^*)$ 更新为 $(upk_{\bar{b}}^*, usk_{\bar{b}}^*, hpk_{\bar{b}}^*, hsk_{\bar{b}}^*, nym_{\bar{b}}^*, cre_{\bar{b}}^*, ASK_{\bar{b}}^*, S_{\bar{b}}^*, c_{\bar{b}}^*, count^* + 1)$，将表目 $(upk_{1-\bar{b}}^*, usk_{1-\bar{b}}^*, hpk_{1-\bar{b}}^*, hsk_{1-\bar{b}}^*, nym_{1-\bar{b}}^*, cre_{1-\bar{b}}^*, ASK_{1-\bar{b}}^*, S_{1-\bar{b}}^*, c_{1-\bar{b}}^*, count^*)$ 更新为 $(upk_{1-\bar{b}}^*, usk_{1-\bar{b}}^*, hpk_{1-\bar{b}}^*, hsk_{1-\bar{b}}^*, nym_{1-\bar{b}}^*, cre_{1-\bar{b}}^*, S_{1-\bar{b}}^*, c_{1-\bar{b}}^*, count^* + 1)$。在询问 2 阶段，允许 $\mathfrak{A}$ 继续提出询问 1 阶段中的各类询问，但不允许它提出询问 $Corrupt(nym_0^*)$ 或 $Corrupt(nym_1^*)$。

最终，$\mathfrak{A}$ 输出对 $\bar{b}$ 的猜测结果 $\bar{b}'$。若 $\bar{b} = \bar{b}' = 0$，则 $\mathfrak{S}$ 判定 $\bar{Z} = \hat{e}(g,g)^{ab\varsigma}$。若 $\bar{b} = \bar{b}' = 1$，则 $\mathfrak{S}$ 判定 $\bar{R} = \hat{e}(g,g)^{ab\varsigma}$。否则，$\mathfrak{S}$ 输出 failure，即求解给定的 DBDH 问题实例失败。

模式 2　$\mathfrak{S}$ 以 DBDHI 问题实例 $\theta = (\bar{P}, \bar{P}^{\upsilon}, \bar{P}^{\upsilon^2}, \cdots, \bar{P}^{\upsilon^q}, \bar{J}) \in G_0^{q+1} \times G_T$ 作为输入，其目标是对以下两种情况做出分辨：1）$\theta \in \bar{S}_1 = \{(\bar{P}, \bar{P}^{\upsilon}, \bar{P}^{\upsilon^2}, \cdots, \bar{P}^{\upsilon^q}, \hat{e}(\bar{P},\bar{P})^{\frac{1}{\upsilon}} \mid \upsilon \in \mathbb{Z}_p^*\}$；2）$\theta \in \bar{S}_2 = \{(\bar{P}, \bar{P}^{\upsilon}, \bar{P}^{\upsilon^2}, \cdots, \bar{P}^{\upsilon^q}, \bar{R}) \mid \upsilon \in \mathbb{Z}_p^*, \bar{R} \in_R G_T\}$。在初始阶段，$\mathfrak{S}$ 选取 $\alpha \in_R \{1, \cdots, m\}$，且 m 的含义同模式 1。$\mathfrak{S}$ 设置 $n < q$，设置 $y_\alpha = \upsilon - n$。$\mathfrak{S}$ 设置 $\bar{F} = y_\alpha \prod_{i=1}^{n-1} (y_\alpha + \upsilon) = \sum_{i=0}^{n} \bar{A}_i \upsilon^i$, $\tilde{g} = \hat{e}(\bar{P},\bar{P})^{\bar{F}} = \prod_{i=0}^{n} \hat{e}(\bar{P}, \bar{P}^{\upsilon^i})^{\bar{A}_i}$，并且构造集合 $\{\tilde{g}^{\frac{1}{y_\alpha+2}}, \tilde{g}^{\frac{1}{y_\alpha+3}}, ..., \tilde{g}^{\frac{1}{y_\alpha+n-1}}\}$。然后，$\mathfrak{S}$ 采用 *TSetup* 算法中的方式产生 *TPK* 中的剩余元素。$\mathfrak{S}$ 执行 *ASetup* 算法，产生 MSK, APK。$\mathfrak{S}$ 初始化列表 $\mathfrak{L}, \mathfrak{L}_K, \mathfrak{L}_2, \mathfrak{L}_3$，并且向 $\mathfrak{A}$ 发送 TPK, MSK, APK。在询问 1 阶段，$\mathfrak{S}$ 采用以下方式对各类预言机进行模拟。

-$H_K(m), H_2(m), H_3(m)$ 模拟方式同模式 1。

-*USetup*() 对该询问的模拟分以下 2 种情况。

情况 1　若当前为第 α 次询问，则 $\mathfrak{S}$ 选取 $usk_\alpha = f_\alpha \in_R \mathbb{Z}_p^*$，设置 $upk_\alpha = F_\alpha = h_1^{f_\alpha}$。$\mathfrak{S}$ 选取 $hpk_\alpha \in_R G_0$，设置 $hsk_\alpha = \log_{h_2} hpk_\alpha$，并且返回 upk_α, hpk_α。同时，$\mathfrak{S}$ 执行 $\mathfrak{L} \leftarrow \mathfrak{L} \cup (upk_\alpha, usk_\alpha, hpk_\alpha, hsk_\alpha, *, *, *, *, *, *)$。显然，$hsk_\alpha$ 为未知元素。

情况 2　若当前并非第 α 次询问，$\mathfrak{S}$ 选取 $f, y \in_R \mathbb{Z}_p^*$，设置 $upk = F = h_1^f, hpk = Y = h_2^y$，并且返回 upk, hpk。同时，$\mathfrak{S}$ 执行 $\mathfrak{L} \leftarrow \mathfrak{L} \cup (upk, usk, hpk, hsk, *, *, *, *, *, *)$。

-*AttGen*(S, upk) 对该询问的模拟分以下 2 种情况。

情况 1　若 $upk = upk_\alpha$，则 $\mathfrak{S}$ 采用模拟方式产生知识签名 π_1。在 *AttGen* 协议结束后，$\mathfrak{S}$ 获得由 $\mathfrak{A}$ 产生的 $nym_\alpha, cre_\alpha, ASK_\alpha$。$\mathfrak{S}$ 将用户 upk_α 在 $\mathfrak{L}$ 中的对应表目更新为 $(upk_\alpha, usk_\alpha, hpk_\alpha, hsk_\alpha, nym_\alpha, cre_\alpha, ASK_\alpha, S, c_\alpha = 0, count = 0)$。

情况 2　若 $upk \neq upk_\alpha$，则 $\mathfrak{S}$ 根据 upk 在 $\mathfrak{L}$ 中找到对应的表目，并利用 usk, hsk 以诚实方式产生知识签名 π_1。剩余操作同情况 1。

-*Corrupt*(nym) $\mathfrak{S}$ 根据 upk 在 $\mathfrak{L}$ 中找到对应的表目 $(upk, usk, hpk, hsk, nym, cre, ASK, S, c, count)$，若 $upk = upk_\alpha$，则 $\mathfrak{S}$ 模拟失败。否则，若 $c = 0$，$\mathfrak{S}$ 返回 usk, hsk，并且将该表目更新为 $(upk, usk, hpk, hsk, nym, cre, ASK, S, c = 1, count)$。

-*Auth*(Cha, nym). $\mathfrak{S}$ 根据 nym 在 $\mathfrak{L}$ 中找到表目 $(upk, usk, hpk, hsk, nym, cre, ASK, S, c, count)$。同时，$\mathfrak{S}$ 从密文 Cha 中分离出访问结构 $(\boldsymbol{M}, \rho)$，并且检查是否满足 $S \models (\boldsymbol{M}, \rho)$ 且 $count < n$。若是，则 $\mathfrak{S}$ 利用 ASK 从密文 Cha 中恢复出秘密元素 K_{SP}。此后的模拟过程分以下 2 种情况。

情况 1　若 $nym = nym_\alpha$，则 $\mathfrak{S}$ 在集合 $\{\tilde{g}^{\frac{1}{y_\alpha+2}}, \tilde{g}^{\frac{1}{y_\alpha+3}}, \cdots, \tilde{g}^{\frac{1}{y_\alpha+n-1}}\}$ 中选取一个未曾使用的元素，并将其作为 J_k。$\mathfrak{S}$ 选取 $\tilde{A}_\alpha, \tilde{o}_k, E_3 \in_R G_0$，选取 $B, K \in_R G_T$，并且采用与模式 1 下 *Auth* 询问情况 1 中的方式模拟产生元素 E_1, E_2 和知识签名 π_3。最后，$\mathfrak{S}$ 向 $\mathfrak{A}$ 发送 $\sigma = (\pi_3, J_k, \tilde{A}_\alpha, B, K, \tilde{o}_k, E_1, E_2, E_3)$。

情况 2　若 $nym \neq nym_\alpha$，则 $\mathfrak{S}$ 采用诚实方式产生知识签名 π_3，并且向 $\mathfrak{A}$ 返回 $\sigma = (\pi_3, J_k, \tilde{A}_i, B, K, \tilde{o}_k, E_1, E_2, E_3)$。

无论属于哪种情况，$\mathfrak{S}$ 都需要在认证过程结束后将表目 $(upk,usk,hpk,hsk,nym,cre,ASK,\ S,c,count)$ 更新为 $(upk,usk,hpk,hsk,nym,cre,ASK,S,c,count+1)$。

在询问 1 阶段的末尾，$\mathfrak{A}$ 输出 (nym_0^*,nym_1^*,Cha^*)。$\mathfrak{S}$ 检查是否满足 $nym_\alpha \in \{nym_0^*, nym_1^*\}$。若否，则 $\mathfrak{S}$ 模拟失败。同时，$\mathfrak{S}$ 从 Cha^* 中分离出访问结构 $(\boldsymbol{M}^*,\rho^*)$。接下来，$\mathfrak{S}$ 根据 nym_α 从 $\mathfrak{L}$ 中找到对应表目 $(upk_\alpha,usk_\alpha,hpk_\alpha,hsk_\alpha,nym_\alpha,\ cre_\alpha,ASK_\alpha,S_\alpha,c_\alpha,count)$，并且检查是否满足 $S_\alpha \models (\boldsymbol{M}^*,\rho^*)$ 且 $count<n$。若是，则 $\mathfrak{S}$ 采用 *Auth* 询问情况 1 中的方式以用户（$nym_{\overline{b}}^*(\overline{b}\in\{0,1\})$）的身份与 $\mathfrak{A}$ 执行 *Auth* 协议。为此，$\mathfrak{S}$ 需要设置 $J_k=\hat{e}(\overline{P},\overline{P})^{\sum_{i=1}^{n}\overline{\lambda}_i v^{i-1}}\overline{R}^{\overline{\lambda}_0}$，且剩余的模拟方式同 *Auth* 询问的情况 1。在该协议结束后，$\mathfrak{S}$ 将 $\mathfrak{L}$ 中的表目 $(upk_{\overline{b}}^*,usk_{\overline{b}}^*,hpk_{\overline{b}}^*,\ hsk_{\overline{b}}^*,nym_{\overline{b}}^*,cre_{\overline{b}}^*,ASK_{\overline{b}}^*,S_{\overline{b}}^*,c_{\overline{b}}^*,count^*)$ 更新为 $(upk_{\overline{b}}^*,usk_{\overline{b}}^*,hpk_{\overline{b}}^*,hsk_{\overline{b}}^*,nym_{\overline{b}}^*,cre_{\overline{b}}^*,ASK_{\overline{b}}^*,S_{\overline{b}}^*,c_{\overline{b}}^*,count^*+1)$，将表目 $(upk_{1-\overline{b}}^*,usk_{1-\overline{b}}^*,hpk_{1-\overline{b}}^*,hsk_{1-\overline{b}}^*,nym_{1-\overline{b}}^*,cre_{1-\overline{b}}^*,ASK_{1-\overline{b}}^*,S_{1-\overline{b}}^*,c_{1-\overline{b}}^*,count^*)$ 更新为 $(upk_{1-\overline{b}}^*,usk_{1-\overline{b}}^*,hpk_{1-\overline{b}}^*,hsk_{1-\overline{b}}^*,nym_{1-\overline{b}}^*,cre_{1-\overline{b}}^*,S_{1-\overline{b}}^*,c_{1-\overline{b}}^*,count^*+1)$。在询问 2 阶段，允许 $\mathfrak{A}$ 继续提出询问 1 阶段中的各类询问，但不允许它提出询问 $Corrupt(nym_0^*)$ 或 $Corrupt(nym_1^*)$。最终，$\mathfrak{A}$ 输出 $\overline{b}'$。若 $\overline{b}'=\overline{b}$，则 $\mathfrak{S}$ 判定 $\theta\in\overline{S}_1$。否则，$\mathfrak{S}$ 判定 $\theta\in\overline{S}_2$。

可靠性 在初始化阶段，$\mathfrak{S}$ 执行 *TSetup* 算法，产生 *TPK*。同时，$\mathfrak{S}$ 执行 *ASetup* 算法，产生 MSK,APK。$\mathfrak{S}$ 初始化列表 $\mathfrak{L},\mathfrak{L}_K,\mathfrak{L}_2,\mathfrak{L}_3$，并且向 $\mathfrak{A}$ 发送 TPK,APK。在询问阶段，$\mathfrak{S}$ 为 $\mathfrak{A}$ 提供以下类型的预言服务。

-$H_K(m),H_2(m),H_3(m)$ 模拟方式同隐私性实验。

-*USetup*() 模拟方式同隐私性实验模式 1 下的情况 3。

-*AttGen*(S,upk) $\mathfrak{S}$ 以诚实 *AA* 的身份与 $\mathfrak{A}$ 执行 *AttGen* 协议，并且返回 nym,cre,ASK。

-*Corrupt*(nym) $\mathfrak{S}$ 根据 nym 在 $\mathfrak{L}$ 中找到对应表目 $(upk,usk,hpk,hsk,nym,cre,ASK,S,c,count)$。

若 $c=0$，则返回 usk,hsk，并将该表目更新为 $(upk,usk,hpk,hsk,nym,cre,ASK,S,c=1,count)$。

-*Auth*(Cha,nym) 模拟方式同隐私性实验模式 1 下的情况 3。

在该阶段的末尾，$\mathfrak{A}$ 输出 $nym^*,(\boldsymbol{M}^*,\rho^*)$。$\mathfrak{S}$ 检查在 $\mathfrak{L}$ 中是否存在与 nym^* 对应的表目。若是，则以诚实 *SP* 的身份与 $\mathfrak{A}$ 执行 $n+1$ 次 *Auth* 协议，并且根据 $(\boldsymbol{M}^*,\rho^*)$ 产生底层 CP-ABE 方案的挑战密文。最终，$\mathfrak{S}$ 获得 $\mathfrak{A}$ 在这些协议中的输出 $\sigma_1,\sigma_2,\cdots,\sigma_{n+1}$。假设这 $n+1$ 次执行过程都获得成功，显然 $\mathfrak{S}$ 可以利用重绕技术分别从知识签名 $\pi_{3,1},\pi_{3,2},\cdots,\pi_{3,n+1}$ 中提取出秘密元素 $k_1^*,k_2^*,\cdots,k_{n+1}^*$ 且它们都位于区间 $[1,n]$。根据底层区间证明协议的有效性，显然存在 $i,j\in[1,n]$，使 $k_i^*=k_j^*=k^*$，即 $\mathfrak{A}$ 在第 i 次和第 j 次 *Auth* 协议中使用了相同的令牌 $J_{k^*}=\tilde{g}^{\frac{1}{k^*+y+1}}$。由于 $\mathfrak{S}$ 充当诚实 *SP*，因此这两次协议不可能都获得成功，即得到矛盾假设。

抗合谋攻击 $\mathfrak{S}$ 以底层 CP-ABE 方案的公钥 $(\overline{g}^\alpha,\hat{e}(\overline{g},\overline{g})^\beta,\{PK_j\}_{j\in U})$ 为输入，其目标是借助与 $\mathfrak{A}$ 的交互过程攻破该方案的选择性 CPA 安全性。在初始化阶段，$\mathfrak{A}$ 首先向 $\mathfrak{S}$ 提供访问结构 $(\boldsymbol{M}^*,\rho^*)$。$\mathfrak{S}$ 设置 $g=\overline{g}$，并且采用 *TSetup* 和 *ASetup* 算法中的方式产生 *TPK* 和 *APK* 中的剩余参数。最后，$\mathfrak{S}$ 初始化列表 $\mathfrak{L},\mathfrak{L}_{nym},\mathfrak{L}_K,\mathfrak{L}_2,\mathfrak{L}_3$，并且向 $\mathfrak{A}$ 发送 TPK,APK。在询问 1 阶段，$\mathfrak{S}$ 为 $\mathfrak{A}$ 提供以下类型的预言服务。

-$H_K(m),H_2(m),H_3(m)$ 模拟方式同隐私性实验。

-*USetup*() 模拟方式同隐私性实验模式 1 下的情况 3。

-*AttGen*(S,upk) $\mathfrak{S}$ 以 *AA* 的身份与 $\mathfrak{A}$ 执行 *AttGen* 协议。$\mathfrak{S}$ 根据 upk 在 $\mathfrak{L}$ 中找到对应的表目 $(upk,usk,hpk,hsk,*,*,*,*,*,*,*)$。$\mathfrak{S}$ 检查是否满足 $S\models(\boldsymbol{M}^*,\rho^*)$，若是，则借助底层 CP-ABE 方案的密钥产生预言机为该用户产生 *ASK*，并且自行为该用户产生 $nym,cre=(A,x)$。$\mathfrak{S}$ 向 $\mathfrak{A}$ 发送 nym,cre,ASK，并且将表目 $(upk,usk,hpk,hsk,*,*,*,*,*,*,*)$ 更新为 $(upk,usk,hpk,hsk,nym,\ cre,ASK,S,c=0,count=0)$。同时，$\mathfrak{S}$ 设置 $\mathfrak{L}_{nym}\leftarrow\mathfrak{L}_{nym}\cup(nym)$。

-*AUpdate*(S,j,w) $\mathfrak{S}$ 检查是否满足 $S\models(\boldsymbol{M}^*,\rho^*)$ 且将属性 j 更新为 w 后，更新后的属性集合 S' 是否满足 $S'\models(\boldsymbol{M}^*,\rho^*)$。若是，则 $\mathfrak{S}$ 借助底层 CP-ABE 方案的属性更新预言机为 $\mathfrak{A}$ 产生 $UK_{j\to w}$，并且对 *APK* 中的元素 PK_j 以及 $\mathfrak{L}$ 中其他用户属性私钥中的元素 D_j,D_j' 进行更新。

-*Corrupt*(nym) 模拟方式同可靠性实验。

-*Auth*(Cha,nym) 模拟方式同隐私性实验模式 1

下的情况 3。

在该阶段的末尾，$\mathfrak{A}$ 输出 K_{SP_0}, K_{SP_1}, NYM。$\mathfrak{S}$ 检查是否满足 $NYM \subseteq \mathfrak{L}_{nym}$，若是，则以 SP 的身份与 $\mathfrak{A}$ 执行 *Auth* 协议。在 *Auth* 协议执行过程中，$\mathfrak{S}$ 向底层 CP-ABE 方案的加密预言机发送 K_{SP_0}, K_{SP_1}, RL，并且向 $\mathfrak{A}$ 返回由该预言机产生的关于元素 $K_{SP_{\bar{b}}}(\bar{b} \in \{0,1\})$ 的密文 $Cha^* = ((\boldsymbol{M}^*, \rho^*), \tilde{C}, C, \{C_i, C_i', C_i''\}_{i=1}^{l}, \{C_{1,t}^*, C_{2,t}^*\}_{t=1}^{|RL|})$。在询问 2 阶段，允许 $\mathfrak{A}$ 继续提出上述类型的询问。

但是，不允许 $\mathfrak{A}$ 利用 *AUpdate* 询问获得能满足访问结构 $(\boldsymbol{M}^*, \rho^*)$ 的属性私钥。同时，不允许 $\mathfrak{A}$ 提出 *AttGen* 询问，使得对于所产生的 nym，满足 $nym \notin NYM$。最终，$\mathfrak{A}$ 输出猜测结果 $\bar{b}'$。若 $\bar{b}' = \bar{b}$，则表明 $\mathfrak{S}$ 已经借助 $\mathfrak{A}$ 攻破了底层 CP-ABE 方案的选择性 CPA 安全性。

可追踪性　当前实验的执行过程分以下 2 种模式。

模式 1　$\mathfrak{S}$ 以底层 CL-SDH 签名方案[6]的公钥 $(\bar{g}, \bar{g}_1, \bar{g}_2, \bar{g}_1^{\bar{\gamma}}) \in G_0^4$ 作为输入，其目标是借助与 $\mathfrak{A}$ 的交互过程攻破该方案的不可伪造性。$\mathfrak{S}$ 设置 $g = \bar{g}, h_1 = \bar{g}_1, h_2 = \bar{g}_2, w_1 = \bar{g}_1^{\bar{\gamma}}, \gamma_1 = \bar{\gamma}$，并且采用 *TSetup*, *ASetup* 算法中的方式产生 TPK, MSK, APK 中的剩余参数。$\mathfrak{S}$ 初始化列表 $\mathfrak{L}, \mathfrak{L}_{\text{auth}}, \mathfrak{L}_K, \mathfrak{L}_2, \mathfrak{L}_3$，并且向 $\mathfrak{A}$ 发送 TPK, APK。在询问阶段，允许 $\mathfrak{A}$ 提出以下类型的预言询问。

-$H_K(m), H_2(m), H_3(m)$ 模拟方式同隐私性实验。

-*USetup*() 对当前询问的模拟分以下 2 种情况。在情况 1 中，$\mathfrak{S}$ 为用户选取 $usk = f \in_{\mathrm{R}} \mathbb{Z}_p^*$, $hsk = y \in_{\mathrm{R}} \mathbb{Z}_p^*$，设置 $upk = h_1^f, hpk = h_2^y$。$\mathfrak{S}$ 执行 $\mathfrak{L} \leftarrow \mathfrak{L} \cup (upk, usk, hpk, hsk, *, *, *, *, c = 0, *)$，并且返回 upk, hpk。在情况 2 中，$\mathfrak{S}$ 接收由 $\mathfrak{A}$ 产生的 $upk = F, hpk = Y, \pi_1$。$\mathfrak{S}$ 采用重绕技术从 π_1 中提取出 usk, hsk，并且执行 $\mathfrak{L} \leftarrow \mathfrak{L} \cup (upk, usk, hpk, hsk, *, *, *, *, c = 1, *)$。

-*AttGen*(S, upk) $\mathfrak{S}$ 根据 upk 在 $\mathfrak{L}$ 中找到对应的表目 $(upk, usk, hpk, hsk, *, *, *, *, c, *)$。$\mathfrak{S}$ 通过调用预言机 $H_2(upk)$ 产生 nym。$\mathfrak{S}$ 自行产生 ASK，向底层 CL-SDH 签名预言机提出询问 (upk, hpk)，并且获得后者返回的 $cert = (A, x)$。$\mathfrak{S}$ 返回 nym, cre, ASK，并将该表目更新为 $(upk, usk, hpk, hsk, nym, cre, ASK, S, c, count = 0)$。

-*Corrupt*(nym) 模拟方式同可靠性实验。此外，$\mathfrak{S}$ 执行 $RL \leftarrow RL \cup (usk, nym)$。

-*Auth*(Cha, nym) 模拟方式同隐私性实验模式 1 下的情况 3。此外，$\mathfrak{S}$ 额外设置 $\mathfrak{L}_{Auth} \leftarrow \mathfrak{L}_{Auth} \cup (nym, \sigma)$。

-*Semi-Auth*(Cha, nym) $\mathfrak{S}$ 以诚实 TPM 的身份与 $\mathfrak{A}$（充当 Host）联合产生 *User* 在 *Auth* 协议中的输出。在该过程中，$\mathfrak{S}$ 接收到 $\mathfrak{A}$ 发送的数据，并且返回 R_{2t}, π_3。同时，$\mathfrak{S}$ 执行 $\mathfrak{L}_{Auth} \leftarrow \mathfrak{L}_{Auth} \cup (nym, \sigma)$。

最终，$\mathfrak{A}$ 输出 $nym^*, (\boldsymbol{M}^*, \rho^*)$。$\mathfrak{S}$ 以 SP 的身份与 $\mathfrak{A}$ 执行 *Auth* 协议，并且利用 $(\boldsymbol{M}^*, \rho^*)$ 产生底层 CP-ABE 方案挑战密文 Cha^*。最终，$\mathfrak{S}$ 获得 $\mathfrak{A}$ 的输出 σ^*。若 σ^* 能通过验证，则 $\mathfrak{S}$ 检查是否满足 $(nym^*, \sigma^*) \notin \mathfrak{L}_{Auth}$。若是，则 $\mathfrak{S}$ 利用重绕技术从 σ^* 中提取出秘密知识 (A^*, x^*)，从而攻破了底层 CL-SDH 方案的不可伪造性。

模式 2　$\mathfrak{S}$ 以 q-SDH 问题实例 $(P, Q, Q^x, \cdots, Q^{x^q}) \in G_0^{q+2}$ 作为输入，其目标是输出数对 $(e, P^{\frac{q}{x+e}})$。$\mathfrak{S}$ 设置 $h_1 = Q$，选取 $r, r_2 \in_{\mathrm{R}} \mathbb{Z}_p^*$，设置 $g = Q^r, h_2 = Q^{r_2}$。$\mathfrak{S}$ 采用 *TSetup* 算法中的方式产生 TPK 中的剩余元素。$\mathfrak{S}$ 初始化列表 $\mathfrak{L}, \mathfrak{L}_{\text{auth}}, \mathfrak{L}_K, \mathfrak{L}_2, \mathfrak{L}_3$，向 $\mathfrak{A}$ 发送 TPK，并且获得后者产生的 APK。此外，$\mathfrak{S}$ 事先选取 $\alpha \in_{\mathrm{R}} \{1, \cdots, m\}$，其中，$m$ 的含义同隐私性实验。在询问阶段，允许 $\mathfrak{A}$ 提出以下类型的预言询问。

-$H_K(m), H_2(m), H_3(m)$ 模拟方式同隐私性实验。

-*USetup*() 对当前询问的模拟分以下 2 种情况。情况 1：若当前是第 α 次询问，$\mathfrak{S}$ 设置 $upk_\alpha = Q^x$, $usk_\alpha = x$。其中，usk_α 为未知元素。$\mathfrak{S}$ 选取 $y \in_{\mathrm{R}} \mathbb{Z}_p^*$，设置 $hpk_\alpha = h_2^y, hsk_\alpha = y$。$\mathfrak{S}$ 返回 upk_α, hpk_α，并且设置 $\mathfrak{L} \leftarrow \mathfrak{L} \cup (upk_\alpha, usk_\alpha, hpk_\alpha, hsk_\alpha, *, *, *, *, c = 0, *)$。情况 2：若当前并非第 α 次询问，$\mathfrak{S}$ 采用诚实方式产生 upk, usk, hpk, hsk，返回 upk, hpk，并且设置 $\mathfrak{L} \leftarrow \mathfrak{L} \cup (upk, usk, hpk, hsk, *, *, *, *, c = 0, *)$。

-*AttGen*(S, upk) 对当前询问的模拟分以下 2 种情况。情况 1：若 $upk = upk_\alpha$，则 $\mathfrak{S}$ 采用模拟方式产生 π_1，并获得 $\mathfrak{A}$ 产生的 $nym_\alpha, cert_\alpha, ASK_\alpha$。$\mathfrak{S}$ 将表目 $(upk_\alpha, usk_\alpha, hpk_\alpha, hsk_\alpha, *, *, *, *, c = 0, *)$ 更新为 $(upk_\alpha, usk_\alpha, hpk_\alpha, hsk_\alpha, nym_\alpha, cert_\alpha, ASK_\alpha, S, c = 0, count = 0)$。情况 2：若 $upk \neq upk_\alpha$，则 $\mathfrak{S}$ 采用诚实方式产生 π_1，

并且获得 $\mathfrak{A}$ 产生的 $nym, cert, ASK$ 。$\mathfrak{S}$ 将表目 $(upk, usk, hpk, hsk, *, *, *, *, c=0, *)$ 更新为 $(upk, usk, hpk, hsk, nym, cert, ASK, S, c=0, count=0)$。

-Corrupt(nym) 若 $nym = nym_\alpha$，则 $\mathfrak{S}$ 模拟失败。否则，模拟方式同当前实验的模式 1。

-Auth(Cha, nym) 若 $nym = nym_\alpha$，则 $\mathfrak{S}$ 在不掌握 usk_α 的条件下模拟产生 σ。若 $nym \neq nym_\alpha$，则 $\mathfrak{S}$ 采用诚实方式产生 σ。剩余模拟方式同隐私性实验模式 1 下的情况 3。无论属于哪种情况，$\mathfrak{S}$ 都额外设置 $\mathfrak{L}_{Auth} \leftarrow \mathfrak{L}_{Auth} \cup (nym, \sigma)$。

-Semi-Auth(Cha, nym) $\mathfrak{S}$ 采用模式 1 下 *Semi-Auth* 询问中的方式进行模拟。区别在于，当 $nym = nym_\alpha$ 时，$\mathfrak{S}$ 采用模拟方式产生 R_{2t}, π_3。否则，$\mathfrak{S}$ 采用诚实方式产生 R_{2t}, π_3。

最终，$\mathfrak{A}$ 输出 $nym^*, (\boldsymbol{M}^*, \rho^*)$。$\mathfrak{S}$ 以 *SP* 的身份与 $\mathfrak{A}$ 执行 *Auth* 协议，并且利用 $(\boldsymbol{M}^*, \rho^*)$ 产生底层 CP-ABE 方案的挑战密文 Cha^*。最终，$\mathfrak{S}$ 获得 $\mathfrak{A}$ 的输出 σ^*。若 σ^* 能通过验证，则 $\mathfrak{S}$ 检查是否满足 $(nym^*, \sigma^*) \notin \mathfrak{L}_{Auth}$。若是，则 $\mathfrak{S}$ 利用重绕技术从 σ^* 中提取出秘密知识 f^*。$\mathfrak{S}$ 检查是否满足 $f^* \in \mathfrak{L}$。若否，则 $\mathfrak{S}$ 模拟失败。在 $\frac{1}{m}$ 的概率下，f^* 恰好等于 x。此时，$\mathfrak{S}$ 选取 $e \in_{\mathrm{R}} \mathbb{Z}_p$，输出 $(P'^{\frac{1}{x+e}}, e)$，从而求解了给定的 q-SDH 问题实例。

证毕。

需要指出的是，本文第 6 节改进方案与第 4 节方案的唯一区别是，利用服务器外包解密技术对底层 CP-ABE 方案进行改进。为了证明改进方案的安全性，仅需证明所提出的外包解密版本不会降低原有 CP-ABE 方案的安全性即可。

结论 1 第 4 节方案中采用的底层 CP-ABE 方案满足选择性 CPA 安全性[19]。

定理 3 在随机预言模型下，可以证明第 6 节的底层 CP-ABE 方案外包解密版本满足选择性 RCCA 安全性和可验证性。

证明 选择性 RCCA 安全性。假设模拟器 $\mathfrak{S}$ 以底层 CP-ABE 方案公钥 $(g^\alpha, \hat{e}(g,g)^\beta, \{PK_j\}_{j \in U})$ 为输入，其目标是通过与敌手 $\mathfrak{A}$ 执行以下的选择性 RCCA 安全性实验攻破该方案的选择性 CPA 安全性。在当前实验中，$\mathfrak{S}$ 需要维护列表 $\mathfrak{L}_c, \mathfrak{L}_d, \mathfrak{L}_4, \mathfrak{L}_5, \mathfrak{L}_6$，从而实现对预言机 *Create*$(\cdot)$, *Corrupt*$(\cdot)$, *Decrypt*$(\cdot,\cdot)$, $H_4(\cdot,\cdot), H_5(\cdot), H_6(\cdot)$ 的模拟。

在初始化阶段，$\mathfrak{A}$ 向 $\mathfrak{S}$ 提供作为挑战的访问结构 $(\boldsymbol{M}^*, \rho^*)$。在系统建立阶段，$\mathfrak{S}$ 向 $\mathfrak{A}$ 提供 $(g^\alpha, \hat{e}(g,g)^\beta, \{PK_j\}_{j \in U})$。$\mathfrak{S}$ 初始化列表 $\mathfrak{L}_c, \mathfrak{L}_d, \mathfrak{L}_4, \mathfrak{L}_5, \mathfrak{L}_6$，并且初始化计数器 $count' \leftarrow 0$。

在询问 1 阶段，$\mathfrak{S}$ 为 $\mathfrak{A}$ 提供以下类型的预言服务。

-$H_4(R, m)$ 若 $(R, m, s) \in \mathfrak{L}_4$，则 $\mathfrak{S}$ 返回 s。否则，$\mathfrak{S}$ 选取 $s \in_{\mathrm{R}} \mathbb{Z}_p^*$，并且返回 s。同时，$\mathfrak{S}$ 执行 $\mathfrak{L}_4 \leftarrow \mathfrak{L}_4 \cup (R, m, s)$。

-$H_5(R)$ 若 $(R, \overline{r}) \in \mathfrak{L}_5$，则 $\mathfrak{S}$ 返回 $\overline{r}$。否则，$\mathfrak{S}$ 选取 $\overline{r} \in_{\mathrm{R}} G_T$，并且返回 $\overline{r}$。同时，$\mathfrak{S}$ 执行 $\mathfrak{L}_5 \leftarrow \mathfrak{L}_5 \cup (R, \overline{r})$。

-$H_6(\overline{r})$ 若 $(\overline{r}, tag) \in \mathfrak{L}_6$，则返回 tag。否则，$\mathfrak{S}$ 选取 $tag \in_{\mathrm{R}} \mathbb{Z}_p^*$，并且返回 tag。同时，$\mathfrak{S}$ 执行 $\mathfrak{L}_6 \leftarrow \mathfrak{L}_6 \cup (\overline{r}, tag)$。

-Create(S) $\mathfrak{S}$ 设置 $count' \leftarrow count' + 1$，并且分以下两种情况进行模拟。

情况 1 若 $S \models (\boldsymbol{M}^*, \rho^*)$，则 $\mathfrak{S}$ 选取 $\hat{\alpha}, \hat{\beta}, \{\hat{v}_j\}_{j \in S}, x \in_{\mathrm{R}} \mathbb{Z}_p^*$，设置 $D = g^{\hat{\beta} + \hat{\alpha} x}, D' = g^{(\hat{\alpha} + \hat{\beta} + \hat{\alpha} x) nym}$，$D'' = D g^{\hat{\alpha}}$。对于每个属性 $j \in S$，$\mathfrak{S}$ 选取 $r_j \in_{\mathrm{R}} \mathbb{Z}_p$，设置 $D_j = g^x H_0(j)^{\hat{v}_j r_j}, D_j' = (g^{\hat{\alpha}})^{r_j}$，$TK = (S, D, D', D'', \{D_j, D_j'\}_{j \in S}), ASK = (\hat{\alpha}, \hat{\beta}, TK)$。

情况 2 若 $S \not\models (\boldsymbol{M}^*, \rho^*)$，$\mathfrak{S}$ 以 (S, x) 为输入调用底层 CP-ABE 方案的密钥产生预言机，并且获得后者返回的 $(\tilde{D}, \tilde{D}', \{\tilde{D}_j, \tilde{D}_j'\}_{j \in S}) = (g^{\beta + \alpha x}, g^{(\alpha + \beta + \alpha x) nym}, \{g^x H_0(j)^{v_j r_j}, (g^\alpha)^{r_j}\}_{j \in S})$。$\mathfrak{S}$ 选取 $z \in_{\mathrm{R}} \mathbb{Z}_p$，设置 $TK = (S, \tilde{D}^{\frac{1}{z}}, \tilde{D}'^{\frac{1}{z}}, (\tilde{D} \cdot g^\alpha)^{\frac{1}{z}}, \{\tilde{D}_j^{\frac{1}{z}}, \tilde{D}_j'^{\frac{1}{z}}\}_{j \in S})$，设置 $ASK = (z, TK)$。

最后，无论属于哪种情况，$\mathfrak{S}$ 均设置 $\mathfrak{L}_c \leftarrow \mathfrak{L}_c \cup (count', S, ASK, TK)$，并且返回 TK。

-Corrupt(i) $\mathfrak{S}$ 根据序号 i 在列表 $\mathfrak{L}_c$ 中找到对应表目 (i, S, ASK, TK)，并且检查是否满足 $S \models (\boldsymbol{M}^*, \rho^*)$。若是，则 $\mathfrak{S}$ 返回 $\perp$。否则，$\mathfrak{S}$ 返回 ASK，并且设置 $\mathfrak{L}_d \leftarrow \mathfrak{L}_d \cup (S)$。

-Decrypt(i, Cha). $\mathfrak{S}$ 根据序号 i 在列表 $\mathfrak{L}_c$ 中找到对应的表目 (i, S, ASK, TK)。然后，$\mathfrak{S}$ 从 Cha 中分离出访问结构 $(\boldsymbol{M}, \rho)$。若 $S \not\models (\boldsymbol{M}, \rho)$，$\mathfrak{S}$ 返回 $\perp$。否则，$\mathfrak{S}$ 分以下两种情况进行模拟。

情况 1 满足 $(\boldsymbol{M}, \rho) \neq (\boldsymbol{M}^*, \rho^*)$。$\mathfrak{S}$ 将 ASK 分

离为 $ASK=(z,TK)$ 的形式，并执行以下步骤。

1) $\mathfrak{S}$ 利用 Cha 和 TK 调用 $Transform_{\text{out}}$ 算法，并得到部分的解密结果 $Cha'=(T_0,T_1,T_2)$。

2) $\mathfrak{S}$ 计算 $R=\dfrac{T_0}{T_2^z}$。

3) $\mathfrak{S}$ 根据 R 在列表 $\mathfrak{L}_4$ 中找到对应的表目 (R,m,s)。根据 R 在列表 $\mathfrak{L}_5$ 中找到对应的表目 $(R,\overline{r})$。同时，根据 $\overline{r}$ 在列表 $\mathfrak{L}_6$ 中找到对应的表目 $(\overline{r},tag)$。若 $\mathfrak{S}$ 无法在 $\mathfrak{L}_4$ 或 $\mathfrak{L}_5$ 或 $\mathfrak{L}_6$ 中找到对应表目，则 $\mathfrak{S}$ 返回 $\perp$。若所找到的匹配表目数量超过 1 个，则 $\mathfrak{S}$ 模拟失败。

4) $\mathfrak{S}$ 验证是否同时满足 $T_0=R(\hat{e}(g,g)^{\beta})^s$，$T_1=m\oplus\overline{r}$，$T_2=(\hat{e}(g,g)^{\beta})^{\frac{s}{z}}$，$tag=H_6(\overline{r})$。若是，则 $\mathfrak{S}$ 返回解密结果 m，否则，$\mathfrak{S}$ 返回 $\perp$。

情况 2　满足 $(\boldsymbol{M},\rho)=(\boldsymbol{M}^*,\rho^*)$。$\mathfrak{S}$ 将 ASK 分离为 $ASK=(\hat{\alpha},\hat{\beta},TK)$ 的形式，并执行以下步骤。

1) $\mathfrak{S}$ 利用 Cha 和 TK 调用 $Transform_{\text{out}}$ 算法，并得到部分的解密结果 $Cha'=(T_0,T_1,T_2)$。此时，满足 $T_0=R\hat{e}(g,g)^{\beta s}$，$T_1=m\oplus\overline{r}$。同时，

$$T_2=\frac{\hat{e}(D,C)\ \prod_{t=1}^{|RL|}\left(\hat{e}(D'',C_{1,t}^*)\ \hat{e}(D',C_{2,t}^*)\right)^{\frac{1}{nym-nym_t^*}}}{\prod_{i\in I}\left(\dfrac{\hat{e}(D_{\rho(i)},C_i^{\omega_i})\ \ \hat{e}(D'',C_i''^{\omega_i})}{\hat{e}(D'_{\rho(i)},C_i'^{\omega_i})}\right)}=\frac{\hat{e}(g^{\hat{\beta}+\hat{\alpha}x},g^s)\prod_{t=1}^{|RL|}\left(\hat{e}(g^{\hat{\alpha}+\hat{\beta}+\hat{\alpha}x},g^{-s_t\cdot nym_t^*})\ \hat{e}(g^{(\hat{\alpha}+\hat{\beta}+\hat{\alpha}x)nym},g^{s_t})\right)^{\frac{1}{nym-nym_t^*}}}{\prod_{i\in I}\left(\dfrac{\hat{e}(g^xH_0(\rho(i))^{\hat{v}_{\rho(i)}r_{\rho(i)}},g^{\hat{\alpha}\boldsymbol{M}_i\vec{v}\omega_i})\ \hat{e}(g^{\hat{\alpha}+\hat{\beta}+\hat{\alpha}x},g^{\boldsymbol{M}_i\vec{v}'\omega_i})}{\hat{e}(g^{\hat{\alpha}r_{\rho(i)}},H_0(\rho(i))^{\hat{v}_{\rho(i)}\boldsymbol{M}_i\vec{v}\omega_i})}\right)}=\hat{e}(g,g)^{\hat{\beta}s}$$

2) $\mathfrak{S}$ 计算 $\overline{T}_2=T_2^{\frac{1}{\hat{\beta}}}$。

3) $\mathfrak{S}$ 在列表 $\mathfrak{L}_4$ 中寻找是否存在元组 (R_i,m_i,s_i)，使 $\overline{T}_2=\hat{e}(g,g)^{s_i}$。若查找失败，则 $\mathfrak{S}$ 返回 $\perp$。若匹配的元组数量超过 1 个，则 $\mathfrak{S}$ 模拟失败。假设 (R,m,s) 为唯一匹配元组。然后，$\mathfrak{S}$ 根据 R 在列表 $\mathfrak{L}_5$ 中寻找匹配的元组。若查找失败，则返回 $\perp$。若匹配的元组数量超过 1 个，则 $\mathfrak{S}$ 模拟失败。假设 $(R,\overline{r})$ 为唯一匹配元组。类似地，$\mathfrak{S}$ 根据 $\overline{r}$ 在列表 $\mathfrak{L}_6$ 中寻找匹配的元组。假设 $(\overline{r},tag)$ 为唯一匹配的元组。

4) $\mathfrak{S}$ 验证是否同时满足 $T_0=R(\hat{e}(g,g)^{\beta})^s$，$T_1=m\oplus\overline{r}$，$T_2=\hat{e}(g,g)^{\hat{\beta}s}$，$tag=H_6(\overline{r})$。若是，则 $\mathfrak{S}$ 返回解密结果 m，否则，$\mathfrak{S}$ 返回 $\perp$。

在挑战阶段，$\mathfrak{A}$ 输出 $K_{SP_0}^*,K_{SP_1}^*$。$\mathfrak{S}$ 选取 $R_0,R_1\in_{\text{R}}G_T$，向底层 CP-ABE 方案的加密预言机发送 $R_0,R_1,(\boldsymbol{M}^*,\rho^*)$，并且获得后者返回的挑战密文 $((\boldsymbol{M}^*,\rho^*),\tilde{C},C,\{C_i,C_i',C_i''\}_{i=1}^{l},\{C_{1,t}^*,C_{2,t}^*\}_{t=1}^{|RL|})$。$\mathfrak{S}$ 选取 $\hat{C}\in_{\text{R}}G_T,tag\in_{\text{R}}\mathbb{Z}_p^*$，并且返回 $Cha=((\boldsymbol{M},\rho),\tilde{C},C,\hat{C},\{C_i,C_i',C_i''\}_{i=1}^{l},\{C_{1,t}^*,C_{2,t}^*\}_{t=1}^{|RL|},tag)$。在询问 2 阶段，允许 $\mathfrak{A}$ 继续提出上述类型的询问。唯一的区别是，当 $Decrypt$ 询问的结果为 $K_{SP_0}^*$ 或 $K_{SP_1}^*$ 时，$\mathfrak{S}$ 返回 $\perp$。最终，$\mathfrak{A}$ 输出猜测结果 $\overline{b}'$。$\mathfrak{S}$ 在列表 $\mathfrak{L}_4$ 与 $\mathfrak{L}_5$ 中检查是否存在以元素 R_0 或 R_1 为首元素的表目。若查找失败，或者存在分别以元素 R_0 和 R_1 为首元素的表目，则 $\mathfrak{S}$ 选取 $\overline{b}''\in_{\text{R}}\{0,1\}$，并将 $\overline{b}''$ 作为自己的猜测结果。否则，假设仅存在以元素 $R_{\overline{b}}(\overline{b}\in\{0,1\})$ 为首元素的匹配表目，则 $\mathfrak{S}$ 将 $\overline{b}$ 作为自己的猜测结果。

可验证性　$\mathfrak{S}$ 采用 $TSetup$ 算法和 $ASetup$ 算法中的方式产生底层 CP-ABE 方案的公钥 APK 和 MSK。在询问 1 阶段，$\mathfrak{S}$ 利用 MSK 实现对预言机 $Create(\cdot),Corrupt(\cdot),Decrypt(\cdot,\cdot)$ 的模拟。在挑战阶段，$\mathfrak{A}$ 输出 $K_{SP}^*,(\boldsymbol{M},\rho)$。此时，$\mathfrak{S}$ 返回 $Cha^*=Encrypt(APK,K_{SP}^*,(\boldsymbol{M},\rho))=((\boldsymbol{M},\rho),\ ((\boldsymbol{M},\rho),\tilde{C},C,\hat{C},\{C_i,C_i',C_i''\}_{i=1}^{l},\{C_{1,t}^*,C_{2,t}^*\}_{t=1}^{|RL|},tag)$。在询问 2 阶段，允许 $\mathfrak{A}$ 继续提出上述类型的询问。最终，$\mathfrak{A}$ 输出 $S^*,cha^{*\prime}=(T_0',T_1',T_2')$。假设 $\mathfrak{S}$ 此前已经产生关于 S^* 的 ASK_{S^*} 和 TK_{S^*}。否则，$\mathfrak{S}$ 自行执行 $Create(S^*)$ 询问，从而产生这些元素。现在，$\mathfrak{S}$ 根据此前产生的 ASK_{S^*} 执行 $Decrypt_{\text{out}}$ 算法，从 $cha^{*\prime}$ 中恢复 $R',\overline{r}'$，$K_{SP}^{*\prime},s'$，满足 $T_0'=R'(\hat{e}(g,\ \ g)^{\beta})^{s'}$，$T_2'=(\hat{e}(g,g)^{\beta})^{\frac{s'}{z}}$，$tag=H_6(\overline{r}')$。若满足 $K_{SP}^{*\prime}\neq K_{SP}^*$，则表明 $\mathfrak{A}$ 成功地进行了欺诈，即 $cha^{*\prime}$ 是关于另一个明文 $K_{SP}^{*\prime}$ 的部分解密结果。$\mathfrak{S}$ 利用 TK_{S^*} 执行 $Transform_{\text{out}}$ 算法，得到关于 Cha^* 的正确的部分解密结果 $\overline{Cha^{*\prime}}=(\overline{T_0'},\overline{T_1'},\overline{T_2'})$，进而恢复得到 $R,\overline{r},K_{SP}^*,s$，满足 $T_0=R(\hat{e}(g,g)^{\beta})^s$，$T_2=(\hat{e}(g,g)^{\beta})^{\frac{s}{z}}$，$tag=H_6(\overline{r})$。于是，$tag=H_6(\overline{r})=H_6(\overline{r}')$，从而违背了散列预言机 $H_6(\cdot)$ 的抗碰撞性。证毕。

8　性能比较与分析

本文提供了对第 6 节方案与相关方案[3-4,8,10-13]

的性能比较。在表 2，对本文方案与这些方案进行了主要性质的比较。本文方案是基于 DAA 技术构造的，因此可以部署于可信平台之上，而其他方案都是在标准 PC 平台上设计的。包括本文方案在内的多数方案都支持可表述性的认证策略（expressive policy）。相反，文献[3,12]方案仅支持受限制的门限认证策略（threshold policy）。文献[11]方案是唯一的不满足属性匿名性的方案，因为该方案要求用户在认证阶段向验证者揭示所使用的属性子集，显然会有损用户的隐私。本文方案和文献[13]方案均采用区间证明技术实现了对用户认证次数的限制，因此较其他方案更适合于按需付费的云服务模式。本文方案满足注册过程的可验证性，使用户可以对所得成员证书和属性私钥进行有效性验证。相反，其他方案[3-4,8,10-13]并未考虑此项验证，即用户必须完全信任 *AA*。此外，本文方案继承了底层 DAA 方案的验证者本地废除机制和底层 CP-ABE 方案的属性更新机制，从而更好地满足了应用系统的实际需求。

在表 3 中，提供了本文方案与相关方案在用户注册（即属性私钥产生）和认证子协议中的通信性能比较。在比较过程中，符号 $|S_i|$ 表示用户属性集合的规模。$|des(\Gamma)|$ 表示由 *SP*（或验证者）定义的属性树（或访问结构）的规模。d 与 k 表示文献[3]中默认属性集合的规模和系统门限值。n' 表示文献[3,12]方案的公开属性集合的规模。l 与 n 分别表示文献[4,8,13]方案和本文方案的访问结构中矩阵 $\boldsymbol{M}$ 的行数和列数，且 l 同样等于文献[10]方案属性树中的叶结点数量[21]。N 表示文献[12]方案中分布式 *AA* 的数量。文献[12]并未提供有关底层两方密钥协商子协议的完整描述，因此用 $|2PC|_U$ 与 $|2PC|_{AA}$ 分别表示 *User* 与 *AA* 在该子协议中的通信开销。$|RL|$ 表示本文方案中被废除的用户数量。对于本文方案，符号“*”标记了 *User* 在外包解密过程中与 *Server* 间的额外通信开销。根据文献[24]结论，在对称的对环境下，域 $\mathbb{Z}_p$ 和群 G_0, G_T 上的元素长度分别为 160 bit，160 bit，960 bit。需要指出的是，文献[3-4,8,10-11]方案的注册阶段不要求 *User* 发送任何数据。原因在于，这些方案假设 *AA* 是完全可信

表 2 相关方案的重要特性总结

方案	计算平台	访问策略	属性匿名性	访问次数受限	注册过程可验证性	成员废除	属性更新
文献[3]	PC	门限	是	否	否	否	否
文献[4]	PC	可表述性	是	否	否	否	否
文献[8]	PC	可表述性	是	否	否	否	否
文献[10]	PC	可表述性	是	否	否	否	否
文献[11]	PC	可表述性	否	否	否	否	否
文献[12]	PC	门限	是	否	否	否	否
文献[13]	PC	可表述性	是	是	否	否	否
本文	可信平台	可表述性	是	是	是	是	是

表 3 相关方案的通信开销比较

方案	注册		认证	
	用户	属性权威/群管理员	用户	服务提供商/验证者
文献[3]	—	$320(\|S_i\|+d)$	$480(n'+d-k+1)+\|des(\Gamma)\|$	—
文献[4]	—	$160\|S_i\|+320$	$160(l+n)+320$	$\|des(\Gamma)\|$
文献[8]	—	$160\|S_i\|+320$	$160(l+n)+320$	$\|des(\Gamma)\|$
文献[10]	—	$160\|S_i\|+320$	1600	$des(\Gamma)\|+320l+160$
文献[11]	—	$160\|S_i\|+320$	$160\|S_i\|+2\,080$	$\|des(\Gamma)\|$
文献[12]	$N(N\text{-}1)\cdot\|2\text{PC}\|_U+640N$	$(N\text{-}1)\cdot\|2\text{PC}\|_{AA}+320\|S_i\|$	$160Nn'+1\,600$	$N\cdot\|des(\Gamma)\|$
文献[13]	480	$160\|S_i\|+480$	$960n+160l+5\,600$	$\|des(\Gamma)\|+320$
本文	800	$480\|S_i\|+1\,280$	$5\,280$ $5\,280+480l+\|des(\Gamma)\|+320(\|RL\|+\|S_i\|)^*$	$\|des(\Gamma)\|+160(3l+\|RL\|)+1120$

的，*User* 直接接收由前者发送的属性私钥。

在表 4 中，提供了本文方案与相关方案在用户注册（即属性私钥产生）和认证子协议中的运算性能比较。对于文献[3, 10]方案，本文采用了 Goyal 等[25]的估算方法，即用 $|S_r|$ 表示用户在基于属性树的认证过程中使用的叶结点集合规模，使对运算次数达到最小。$|L_i|$ 是文献[11]方案中的特殊参数，它表示根据 S_i 构造的属性树的虚拟叶结点集合规模。此外，$(2PC)_U$ 与 $(2PC)_{AA}$ 分别表示 *User* 与 *AA* 在文献[12]方案的底层两方密钥协商子协议中的运算开销。在比较过程中，本文仅统计群 G_0, G_T 上的指数运算和对运算，且认为单指数运算和多指数运算的开销相当。对于本文方案，符号“+”与“++”分别标记了 TPM 和 Host 的运算开销。根据文献[26-27]结论，在对称的对环境下，以群 G_0 上的指数运算为基准（记为 E），群 G_T 上的指数运算开销约为 $4E$，且对运算的开销约为 $3E$。最终，本文将所有方案中各角色的运算开销都换算为群 G_0 上的指数运算，从而得到较为直观的比较结果。显然，本文方案在用户端的认证过程运算效率方面是最优的，而且为常量。取得这个优势的原因是：1) *User* 对挑战密文 *Cha* 进行了外包；2) 在 π_3 的产生过程中，TPM 与 Host 的绝大多数运算都采用预计算方式完成；3) *User* 通过 π_3 向 *SP* 证明“自己掌握合法的成员证书”且“认证令牌 J_k 的产生过程使用了在公开集合 Φ 中选取的新鲜下标 k ”。为了证明上述断言，本文方案采用了无需用户端执行对运算的知识证明技术。

与已有同类方案[3-4,8,10-13]相比，本文方案额外支持属性更新和用户废除。在属性更新协议中，*User* 的运算可以在离线阶段完成，且 *AA* 的在线运算开销为 $(4|RL|+n_{non}+13)E$，其中，n_{non} 表示当某用户的特定属性发生变化后，其他拥有该属性但属性未发生更新的用户数量。在用户废除算法中，*User* 无需执行任何运算，且 *AA* 的运算开销为 $9E$。应当承认的是，本文方案在增加理想性质同时也付出了额外的运算和通信开销。具体地，在注册阶段，*User* 与 *AA* 的通信和运算开销较大，这是为实现可验证的注册性质而付出的必要代价。在认证阶段，*SP* 端的通信和运算开销均与 $|RL|$ 有关，这是为实现用户废除性质而付出的必要代价。此外，为了在最大程度上降低用户端在认证阶段的运算开销，要求 *User* 将挑战密文 *Cha* 和转换钥 *TK* 转发给 *Server*，从而产生了一笔额外的通信开销。因此，在 *AA* 未必完全可信、用户属性集合规模较大且要求支持成员废除和属性更新的现实应用环境下，本文方案具有明显优势。

9　结束语

本文基于 DAA 、支持属性更新的 CP-ABE 和基于成员身份的区间证明等技术提出新的 k-TABA 方案。该方案可部署于可信平台，并且采用预计算技术和服务器外包解密技术尽可能地优化了用户端的运算性能。与 Yuen 等的 k-TABA 方案和相关属性认证方案相比，本文方案不但支持一般性的访问控制策略，而且较好地解决了现实应用中的用户私钥废除和用户属性值更新问题。相对于已有的同类方案，本文方案的显著特点是使得用户在认证协议中的运算耗费摆脱了对底层属性集合和属性树（或访问结构）规模的依赖。

表 4　　**相关方案的运算开销比较**

方案	注册		认证	
	用户	属性权威/群管理员	用户	服务提供商/验证者
文献[3]	—	$2(\|S_i\|+d)E$	$(2(n'+d-k)+5)E$	$\|S_r\|(6(n'+d-k)+8)E$
文献[4]	—	$(\|S_i\|+1)E$	$(3+2l+2ln)E$	$(5ln+3n+6)E$
文献[8]	—	$(\|S_i\|+1)E$	$(3+2l+2ln)E$	$(5ln+3n+6)E$
文献[10]	—	$(\|S_i\|+1)E$	$(6\|S_r\|+14)E$	$(2l+8)E$
文献[11]	—	$(\|S_i\|+1)E$	$(26+2\|S_i\|+\|L_i\|)E$	$(28+2\|S_i\|+\|L_i\|)E$
文献[12]	$N(N\text{-}1)(2PC)_U+NE$	$(N\text{-}1)(2PC)_{AA}+(2\|S_i\|+1)E$	$(N\|S_i\|+2Nn'+5)E$	$(10+3Nn')E$
文献[13]	E	$(\|S_i\|+3)E$	$(32+l+2ln+3n)E$	$(5ln+7n+32)E$
本文	$4E^{+}$ $(2\|S_i\|+13)E^{++}$	$(4\|S_i\|+11)E$	— $12E^{++}$	$(23+4\|RL\|)E$

附录　定理 1 的证明

定理 1　第 4 节方案满足正确性。

证明　为了证明本文方案的正确性，需要分别证明 *AttGen*, *Auth*, *AUpdate* 协议均满足正确性。

AttGen **协议的正确性**　π_1 是标准的“关于掌握离散对数”的知识签名，它使 *AA* 可以确信 TPM 提供的 upk, hpk 满足正确性，即满足 $upk = F = h_1^f, hpk = Y = h_2^y$。若 $cert_i = (A_i, x_i)$ 是采用正确方式产生的，则必然满足

$$A_i = (gFY)^{\frac{1}{\gamma_1 + x_i}} \tag{1}$$

显然，式(1)等价于以下的式(2)、式(3)同时成立，即

$$A_i^{\gamma_1} = A_i^{-x_i} gFY \tag{2}$$

$$w_1 = g^{\gamma_1} \tag{3}$$

同时，若 ASK_i 中的元素 D, D' 是采用正确方式产生的，则分别满足

$$D = g^{\beta + \alpha x_i} \tag{4}$$

$$D' = g^{(\alpha+\beta+\alpha x_i) nym_i} \tag{5}$$

显然，式(4)和式(5)分别等价于以下的式(6)和式(7)，即

$$\hat{e}(D, g) = \hat{e}(g, g)^{\beta}\ \hat{e}(g^{x_i}, g)^{\alpha} \tag{6}$$

$$\hat{e}(D', g) = \hat{e}(D, g)^{nym_i}\ \hat{e}(g^{\alpha}, g)^{nym_i} \tag{7}$$

π_2 是标准的“关于掌握离散对数”的知识签名，它与额外的验证等式 $\hat{e}(D,\ g)\hat{e}(g^{\alpha}, g) = \hat{e}(D', g)^{nym_i^{-1}}$ 可以确保 $cert_i = (A_i, x_i), ASK_i = (S_i, D, D', \{D_j, D_j'\}_{j \in S_i})$ 满足正确性。

Auth 协议的正确性　π_3 是标准的“关于掌握离散对数”的知识签名，它使 *SP* 可以确信 *User* 提供的元素 J_k, K, B 满足正确性，即满足 $J_k = \tilde{g}^{\frac{1}{y+k+1}}, K = B^f$。同时，$\pi_3$ 表明以下等式成立，即

$$E_1 = \tilde{A}_i^{-x_i} g^{\delta} h_1^{\delta f} h_2^{\delta y} \tag{8}$$

$$E_2 = \tilde{o}_k^{-k} g^{l} \tag{9}$$

$$E_3 = g^k h^{\nu} \tag{10}$$

此外，额外的验证等式 $\hat{e}(E_1, g) = \hat{e}(\tilde{A}_i, w_1)$ 与 $\hat{e}(E_2, g) = \hat{e}(\tilde{o}_k, w_2)$ 表明以下关系成立，即

$$E_1 = \tilde{A}_i^{\gamma_1} \tag{11}$$

$$E_2 = \tilde{o}_k^{\gamma_2} \tag{12}$$

显然，根据式(8)和式(11)式，可以得出

$$A_i = (g h_1^f h_2^y)^{\frac{1}{\gamma_1 + x_i}} \tag{13}$$

同时，将式(12)带入式(9)，可以得出

$$\tilde{o}_k = g^{\frac{l}{\gamma_2 + k}} \tag{14}$$

结合约束关系 $\tilde{o}_k \neq 1_{G_0}$，可以得出

$$o_k = \tilde{o}_k^{l^{-1}} = g^{\frac{1}{\gamma_2 + k}} \in \Sigma \tag{15}$$

最后，根据第 5 节中 π_3 的产生和验证过程可知，对于诚实用户 *User*，π_3 能通过 *SP* 验证的条件是他能从挑战密文 *Cha* 中恢复出元素 K_{SP}，并利用该元素产生 π_3 的挑战串 c。对于密文 $Cha = ((\boldsymbol{M}, \rho), \tilde{C}, C, \{C_i, C_i', C_i''\}_{i=1}^{l}, \{C_{1,t}^*, C_{2,t}^*\}_{t=1}^{|RL|})$，显然满足

$$\prod_{i \in I}\left(\frac{\hat{e}(D_{\rho(i)}, C_i^{\omega_i})\,\hat{e}(D'', C_i''^{\omega_i})}{\hat{e}(D'_{\rho(i)}, C_i'^{\omega_i})}\right) =$$

$$\prod_{i \in I}\left(\frac{\hat{e}(g^{x_i} H_0(\rho(i))^{v_{\rho(i)} r_{\rho(i)}}, g^{\alpha \boldsymbol{M}_i \vec{v} \omega_i})\,\hat{e}\,(g^{\alpha+\beta+\alpha x_i}, g^{\boldsymbol{M}_i \vec{v}' \omega_i})}{\hat{e}(g^{\alpha r_{\rho(i)}}, H_0(\rho(i))^{v_{\rho(i)} \boldsymbol{M}_i \vec{v} \omega_i})}\right) =$$

$$\hat{e}(g,g)^{\alpha x_i \cdot \left(\sum_{i \in I} \boldsymbol{M}_i \omega_i\right) \cdot \vec{v}}\ \hat{e}(g,g)^{(\alpha+\beta+\alpha x_i) \cdot \left(\sum_{i \in I} \boldsymbol{M}_i \omega_i\right) \cdot \vec{v}'} =$$

$$\hat{e}(g,g)^{\alpha x_i \cdot (1,0,0,\ldots,0) \cdot \vec{v}}\ \hat{e}(g,g)^{(\alpha+\beta+\alpha x_i) \cdot (1,0,0,\ldots,0) \cdot \vec{v}'} =$$

$$\hat{e}(g,g)^{\alpha x_i s}\ \hat{e}(g,g)^{(\alpha+\beta+\alpha x_i) s^*} \tag{16}$$

$$\hat{e}(D, C) = \hat{e}(g^{\beta + \alpha x_i}, g^s) \tag{17}$$

$$\prod_{t=1}^{|RL|} (\hat{e}(D'', C_{1,t}^*)\ \hat{e}(D', C_{2,t}^*))^{\frac{1}{nym - nym_t^*}} =$$

$$\prod_{t=1}^{|RL|} (\hat{e}(g^{\alpha+\beta+\alpha x_i}, g^{-s_t \cdot nym_t^*})\ \hat{e}(g^{(\alpha+\beta+\alpha x_i) nym}, g^{s_t}))^{\frac{1}{nym - nym_t^*}} =$$

$$\hat{e}(g,g)^{(\alpha+\beta+\alpha x_i) s^*} \tag{18}$$

最后，结合 $\tilde{C} = K_{SP} \cdot \hat{e}(g,g)^{\beta s}$ 和式(16)~式(18)得出，

$$\frac{\tilde{C} \prod_{i \in I}\left(\frac{\hat{e}(D_{\rho(i)}, C_i^{\omega_i})\,\hat{e}(D'', C_i''^{\omega_i})}{\hat{e}(D'_{\rho(i)}, C_i'^{\omega_i})}\right)}{\hat{e}(D, C) \prod_{t=1}^{|RL|} \left(\hat{e}(D'', C_{1,t}^*)\,\hat{e}(D', C_{2,t}^*)\right)^{\frac{1}{nym - nym_t^*}}} = K_{SP} \tag{19}$$

AUpdate 协议的正确性　该协议的正确性可以直接根据底层 CP-ABE 方案属性更新算法的正确性得出。

证毕

参考文献：

[1] 李顺东, 窦家维, 王道顺. 同态加密算法及其在云安全中的应用[J]. 计算机研究与发展, 2015, 52(6): 1378-1388.
LI S D, DOU J W, WANG D S. Survey on homomorphic encryption and its applications to cloud security[J]. Journal of Computer Research and Development, 2015, 52(6): 1378-1388.

[2] 张凯, 马建峰, 李辉, 等.支持高效撤销的多机构属性加密方案[J]. 通信学报, 2017,38(3):83-91.
ZHANG K, MA J F, LI H, et al. Multi-authority attribute-based encryption with efficient revocation[J]. Journal on Communications, 2017, 38(3): 83-91.

[3] ZHOU J, CAO Z F. PSCPA: patient self-controllable privacy-preserving cooperative authentication in distributed m-healthcare systems[R]. IACR Cryptology ePrint Archive: Report 2012/044.

[4] LI M, HUANG X Y, LIU J K, et al. Cooperative attribute-based access control for enterprise computing system[J]. International Journal of

Embedded Systems, 2015,7(3-4):191-202.

[5] CAMENISCH J, DRIJVERS M, HAJNY J. Scalable revocation scheme for anonymous credentials based on n-times unlinkable proofs[C]// the 2016 ACM on Workshop on Privacy in the Electronic Society, WPES 2016. 2016: 123-133.

[6] NGUYEN L. Efficient dynamic k-times anonymous authentication[C]//First International Conference on Cryptology in Vietnam, VIETCRYPT 2006. Hanoi, Vietnam, 2006: 81-98.

[7] 柳欣, 徐秋亮. 不可克隆的动态 k 次匿名认证方案[J]. 通信学报, 2012,33(7):75-89.
LIU X, XU Q L. Unclonable dynamic k-times anonymous authentication[J]. Journal on Communications, 2012,33(7):75-89.

[8] LIAN Y, HUANG X Y, MU Y. SA 3: Self-adaptive anonymous authentication for dynamic authentication policies[J]. Future Generation Computer Systems, 2014, 30(1): 133-139.

[9] MAJI H K, PRABHAKARAN M, ROSULEK M. Attribute-based signatures[C]//The Cryptographers' Track at the RSA Conference 2011, CT-RSA 2011. 2011: 376-392.

[10] YANG H, OLESHCHUK V A. An efficient traceable attribute-based authentication scheme with one-time attribute trees[C]// 20th Nordic Conference, NordSec 2015. 2015: 123-135.

[11] YANG H, OLESHCHUK V A. A dynamic attribute-based authentication scheme[C]//International Conference on Codes, Cryptology, and Information Security, C2SI 2015. 2015: 106-118.

[12] LI J, CHEN X F, HUANG X Y. New attribute-based authentication and its application in anonymous cloud access service[J]. International Journal of Web and Grid Services, 2015, 11(1): 125-141.

[13] YUEN T H, LIU J K, AU M H, et al. K-times attribute-based anonymous access control for cloud computing[J]. IEEE Transactions on Computers, 2015, 64(9): 2595-2608.

[14] BRICKELL E, CHEN L, LI J. Simplified security notions of direct anonymous attestation and a concrete scheme from pairings[J]. International journal of information security, 2009, 8(5): 315-330.

[15] CHEN L. A DAA scheme requiring less TPM resources[C]//5th International Conference on Information Security and Cryptology. 2010: 350-365.

[16] DESMOULINS N, LESCUYER R, SANDERS O, et al. Direct anonymous attestations with dependent basename opening[C]//the International Conference on Cryptology and Network Security. 2014: 206-221.

[17] BRICKELL E, CHEN L, LI J. A static Diffie-Hellman attack on several direct anonymous attestation schemes[C]//4th International Conference on Trusted Systems, INTRUST 2012. 2012: 95-111.

[18] XI L, YANG Y, ZHANG Z F, et al. DAA-related APIs in TPM 2.0 revisited[C]//the International Conference on Trust and Trustworthy Computing, TRUST 2014. 2014: 1-18.

[19] ZHANG P, CHEN Z H, LIANG K T, et al. A cloud-based access control scheme with user revocation and attribute update[C]//the Australasian Conference on Information Security and Privacy, ACISP 2016. 2016: 525-540.

[20] ARFAOUI G, LALANDE J F, TRAORÉ J, et al. A practical set-membership proof for privacy-preserving NFC mobile ticketing[C]//15th International Symposium on Privacy Enhancing Technologies, PETS 2015. Philadelphia. 2015: 25-45.

[21] GREEN M, HOHENBERGER S, WATERS B. Outsourcing the decryption of abe ciphertexts[C]//The USENIX Security Symposium 2011. San Francisco, CA, USA, 2011: 3-18.

[22] BONEH D, BOYEN X. Short signatures without random oracles and the SDH assumption in bilinear groups[J]. Journal of Cryptology, 2008, 21(2): 149-177.

[23] LAI J Z, DENG R H, GUAN C, et al. Attribute-based encryption with verifiable outsourced decryption[J]. IEEE Transactions on Information Forensics and Security, 2013, 8(8): 1343-1354.

[24] CAMENISCH J, HOHENBERGER S, PEDERSEN M Ø. Batch verification of short signatures[J]. Journal of Cryptology, 2012, 25(4): 723-747.

[25] GOYAL V, PANDEY O, SAHAI A, et al. Attribute-based encryption for fine-grained access control of encrypted data[C]//13th ACM conference on Computer and communications security, CCS 2006. Alexandria. 2006: 89-98.

[26] CHEN L, MORRISSEY P, SMART N P. Pairings in trusted computing[C]//International Conference on Pairing-Based Cryptography, Pairing 2008. 2008: 1-17.

[27] LI J, AU M H, SUSILO W, et al. Attribute-based signature and its applications[C]// 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2010. 2010: 60-69.

[作者简介]

柳欣（1978−），男，山东广饶人，博士，山东青年政治学院副教授，主要研究方向为密码学与信息安全。

徐秋亮（1960−），男，山东淄博人，山东大学教授、博士生导师，主要研究方向为密码学与信息安全。

张斌（1975−），男，山东济南人，博士，山东青年政治学院讲师，主要研究方向为密码学与信息安全。

张波（1981−），男，山东德州人，博士，济南大学讲师，主要研究方向为密码学与信息安全。