

自定义密钥的加密去重云存储访问控制方案

谷博伦^{1,2}, 徐子凯^{1,2}, 李卫海^{1,2}, 俞能海^{1,2}

(1. 中国科学技术大学网络空间安全学院, 安徽 合肥 230026; 2. 数字安全安徽省重点实验室, 安徽 合肥 230026)

摘要: 随着互联网的高速发展和应用, 传统的存储资源难以满足日益增长的海量数据存储的需求, 越来越多的用户尝试将他们的数据上传到第三方云服务器进行统一存储。如何在云上同时实现高效的加密重复数据删除和安全的文件共享已成为迫切需要解决的问题。并且, 用户总是喜欢自己设定密码对文件进行加密, 而在有需要时才对加密文件进行共享。据此, 设计了一个确定性分步加密算法, 当两步加密的密钥满足一定关系时, 两步加密可以等效为一次加密。在此基础上提出了一个支持动态访问控制的云存储加密重复数据删除方案, 使用确定性分步加密算法对文件进行加密, 使用密文策略的属性基加密算法对文件密钥进行加密。这不仅使持有相同文件的不同用户均可以灵活地自定义各自的加解密密钥, 而且还可以通过动态的访问控制机制确保安全的文件共享。此外, 访问控制部分能够兼容大多数现有的密文策略属性基加密方案, 甚至允许在不同的属性组中使用不同的密文策略属性基加密方案。安全性分析结果显示, 该方案能够达到当前加密去重范式下的最高安全性。实验和分析结果表明, 该方案能够较好地满足云服务商和用户的实际需求, 具有良好的算法执行效率。

关键词: 加密去重; 自定义密钥; 访问控制; 分步椭圆曲线加密; 确定性改进型最优非对称加密填充

中图分类号: TP393

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2024055

Key-customizable encrypted deduplication with access control for cloud storage

GU Bolun^{1,2}, XU Zikai^{1,2}, LI Weihai^{1,2}, YU Nenghai^{1,2}

1. School of Cyber Science and Technology, University of Science and Technology of China, Hefei 230026, China

2. Anhui Province Key Laboratory of Digital Security, Hefei 230026, China

Abstract: With the rapid development and application of the Internet, traditional storage resources have been found unable to meet the growing demand for massive data storage. An increasing number of users have attempted to upload their data to third-party cloud servers for unified storage. Efficient deduplication and secure file sharing in the cloud have emerged as critical concerns. Moreover, users have always preferred to customize their passwords for

收稿日期: 2024-03-20; 修回日期: 2024-07-09

通信作者: 李卫海, whli@ustc.edu.cn

基金项目: 国家重点研发计划 (2018YFB0804101)

Foundation Item: The National Key R&D Program of China (2018YFB0804101)

引用格式: 谷博伦, 徐子凯, 李卫海, 等. 自定义密钥的加密去重云存储访问控制方案[J]. 网络与信息安全学报, 2024, 10(4): 85-97.

Citation Format: GU B L, XU Z K, LI W H, et al. Key-customizable encrypted deduplication with access control for cloud storage[J]. Chinese Journal of Network and Information Security, 2024, 10(4): 85-97.

encrypting and decrypting files, only sharing encrypted files when necessary. Based on this preference, a deterministic stepwise encryption algorithm was first designed. It was such that when the keys for the two steps of encryption satisfied a certain relationship, the two steps of encryption could be equivalent to a single encryption process. A novel key-customizable encrypted deduplication scheme with access control for cloud storage was proposed, utilizing the deterministic stepwise encryption algorithm to encrypt files and a ciphertext-policy attribute-based encryption algorithm to encrypt file keys. This scheme not only offered the flexibility to customize encryption and decryption keys for different users with the same files, but also ensured secure file sharing through a dynamic access control mechanism. Moreover, the optional access control component was made compatible with the majority of existing ciphertext-policy attribute-based encryption (CP-ABE) schemes, even allowing for different CP-ABE schemes within different attribute groups. Security analysis results show that the proposed scheme achieves the highest level of security under the current encrypted deduplication paradigm. Experimental and analytical results indicate that it effectively meets the practical needs of cloud service providers and users, and also achieves acceptable efficiency.

Keywords: encrypted deduplication, customizable key, access control, stepwise elliptic curve encryption, deterministic optimal asymmetric encryption padding++

0 引言

互联网自诞生以来发展迅速, 积累了大量的用户和数据。然而个人用户的存储空间非常有限, 传统的存储方式难以满足用户日益增长的存储需求。云存储提供了一种高效、经济的解决方案, 允许用户在云服务器上存储和下载文件。在提供海量存储空间的同时, 云服务器可以发现并去除重复数据, 进一步节省存储资源^[1]。研究表明, 在备份应用系统中, 重复数据删除最多可减少 90% 的存储需求, 在标准文件系统中可减少约 50% 的存储需求^[2]。但是, 用户总是担心第三方云服务商的安全性^[3], 他们希望在将文件上传到云服务器之前对其进行加密。如果每个用户都独立选择自己的密钥和加密算法, 云服务器几乎不可能再从密文中区分出重复数据。并且, 由于云存储可以随时随地提供下载服务, 用户可以借此安全地与他人共享自己的文件。因此, 如何在安全共享的同时对加密文件进行重复数据删除成为一个非常值得研究的问题。

Douceur 等^[4]首先提出了一种收敛加密方案, 使用数据的哈希值作为加解密密钥。该方案将相同的明文加密成相同的密文, 允许云服务器使用传统方法对加密数据进行去重。然而, 该方案无法确保语义的安全性, 并容易受到离线穷举攻击。Bellare 等^[5]对收敛加密进行了形式化定义和安全性证明, 并提出了一种新的密码学原语——消息锁加密 (message-locked encryption, MLE)。

随后, 他们引入了 DupLESS 方案^[6], 该方案利用一个不经意伪随机函数从可信第三方派生相同的加解密密钥。然而, 该方案无法防御云服务器的在线穷举攻击。

后续的研究采用更灵活的密钥生成方法。第一个上传者可以确定加解密密钥, 然后使用密码学算法与后来的上传者共享。Liu 等^[7]使用基于密码身份验证的密钥交换协议建立加解密密钥, 使具有相同数据的不同所有者可以获取相同的密钥。Yan 等^[8]提出了使用基于属性的加密来抵御用户的未授权访问, 并对存储在云中的加密数据进行去重。Xiong 等^[9]在一个分层异构环境中提出了基于角色的对称加密的加密去重方案。该方案可以防止隐私泄露和侧信道攻击, 并且可以对多媒体数据进行去重。Kwon 等^[10]引入了支持用户撤销的去重方案, 利用不经意伪随机函数生成加解密密钥, 并利用基于权限的加密来执行访问策略, 从而增强安全性。FCDedup^[11]采用复合阶双线性配对, 实现云级和雾级的加密重复数据删除。Yuan 等^[12]提出了一种基于收敛全或无变换和布隆过滤器随机采样比特的高效重加密安全数据删除方案。经查证, 目前仍然没有加密去重方案允许所有数据上传者选择并使用它们自己的密钥。

有一些方案试图避免使用额外的密钥服务器以进一步简化系统。Liu 等^[7]提出了第一个不需要可信第三方的去重方案。该方案要求相同数据的不同所有者同时在线, 但这在实际应用中几乎不可能实现。Jiang 等^[13]引入了 μ R-MLE 的概念,

采用客户端和服务端交互的方式，构建了静态和动态两种去重决策树，将重复数据检测的时间从线性降低到对数级别。LEVER^[14]是第一个仅通过密码学的双方交互实现的抵御穷举攻击的加密去重方案，但其使用的同态加密在性能上难以令人满意。Zhang等^[15]提出了一种具有基于身份验证访问控制的安全高效的数据去重方案。Guo等^[16]利用椭圆曲线密码设计了一种密钥共享方法，使不同用户只与云服务器交互即可共享一个随机密钥。该方案还能有效管理数据所有权，但只能通过云存储的ID列表进行有限的访问控制。

现有的加密去重方案虽然在安全性和去重粒度上取得了不错的效果，但在便捷性和实用性方面仍然有所欠缺。用户需要执行大量的额外操作，且不能够像日常加解密场景一样自己设定加解密口令。这也是如今加密去重方案仍未被各大云服务商采用的主要原因。

本文提出了一种具有动态访问控制、自定义密钥的云存储加密重复数据删除方案。它允许每一个用户在可信第三方的帮助下自定义自己的加解密密钥，还可以使用密文策略的属性基加密(ciphertext-policy attribute-based encryption, CP-ABE)方案共享文件。本文的主要贡献总结如下。

(1) 自定义密钥的加密去重

本文方案允许每个用户使用自己设置的密钥加解密自己的文件，而不管用户是否是某个文件的初始上传者。密钥生成中心根据用户自定义密钥计算云服务器的重加密密钥。云服务器将接收到的中间密文重加密为存储密文。提出的确定性改进型最优非对称加密填充的分步椭圆曲线加密(stepwise elliptic curve encryption with deterministic optimal asymmetric encryption padding++, SECC-DOAEP++)算法保证了相同的明文总是产生相同的密文。

(2) 动态的访问控制

本文方案使用SECC-DOAEP++算法对文件进行加密，使用CP-ABE算法对文件密钥进行加密，可以同时实现加密重复数据删除和访问控制，并且无论用户是否启用访问控制都不会影响加密去重本身。

(3) 可扩展的属性群

本文方案中的访问控制功能可作用于不同的属性群中，属性群可以使用各自不同的访问控制策略，并在需要时分别初始化。作为额外开销，云服务器只需要为存储的加密文件记录多个文件密钥。

1 OAEP++的分步ECC算法

最优非对称加密填充(optimal asymmetric encryption padding, OAEP)^[17]由Bellare等首次提出，用于在非对称加密前先对消息进行填充。填充的引入使非对称加密方案的安全性可以被严格地证明，在随机谰言机模型下，使用OAEP的非对称加密方案的安全性可以达到适应性选择密文攻击下的不可区分性^[18]。本文将对OAEP的一种改进型变种OAEP++^[19]进行改进。OAEP++的加密过程如图1所示。

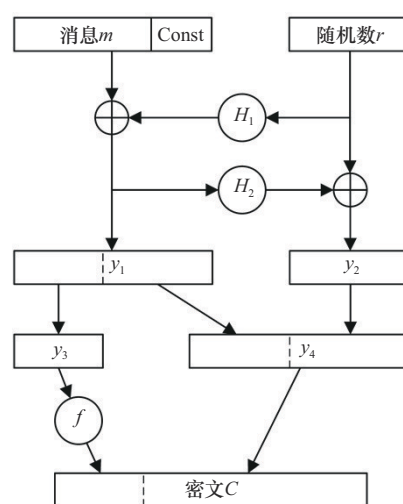


图1 OAEP++的加密过程

Figure 1 Encryption process with OAEP++

最优非对称加密填充的加密过程使用两个伪随机序列发生器 H_1 、 H_2 对消息 m 进行填充，然后使用非对称加密 f （解密函数记为 g ）对填充结果进行加密。解密过程是加密过程的逆。记生成的随机数为 r ，那么加密过程可以表示为：

$$y_1 = (m \parallel \text{Const}) \oplus H_1(r) \quad (1)$$

$$y_2 = r \oplus H_2(y_1) \quad (2)$$

$$y_3 \parallel y_4 = y_1 \parallel y_2 \quad (3)$$

$$C = f(y_3) \parallel y_4 \quad (4)$$

在此基础上, 本文提出了一种新的优化算法称作确定性改进型最优非对称加密填充的分步 ECC (elliptic curve cryptography) 加密 SECC-DOAEP++ (stepwise-ECC with deterministic-OAEP++)。它是 ECC-OAEP++ 的分步和确定性变体, 其中加密和解密过程均由双方分步执行, 并且该分步加密过程可以等效成一次 ECC-DOAEP++ 的加解密过程。在当前的加密去重范式下, SECC-DOAEP++ 能够达到最高的安全级别 D-IND\$-CPA^[20], 这将在安全性分析中详细介绍。

为了满足加密去重的基本要求, 即相同的明文产生相同的密文, 首先将 OAEP++ 转换为确定性填充算法 DOAEP++。OAEP++ 的不确定性来自每次填充过程产生的随机数 r 。简单的思路是直接从消息中 m 确定性地生成随机数 r , 借助伪随机序列发生器 G , 可以得到随机数 r 如下:

$$r = G(m) \quad (5)$$

这种情况下, 相同的消息 m 将总是产生相同的填充结果 M 。值得注意的是, 这种改变不会影响去填充的过程, 仍然可以使用原方法从填充的结果 M 中恢复消息 m 。

然后, 考虑分步 ECC 加密。标准的 ECC 建立在有限域 p 的椭圆曲线 $y^2 = x^3 + ax + b$ 上, 可简记为 $E_p(a, b)$, 且满足 $4a^3 + 27b^2 \neq 0$ 。该椭圆曲线的阶记为 N (在本文中默认椭圆曲线的辅因子为 1, 即 $N = |E_p(a, b)|$, 且阶为素数)。那么, 所有椭圆曲线参数可以记为 $EC = (p, a, b, N)$ 。对于加解密, 为了实现分步加密的效果, 使用 ECC 的一个简化版本 ECC-Massey-Omura, 如无特别说明, 后文中出现的 ECC 加解密均是指 ECC-Massey-Omura 加解密。具体来说, 在椭圆曲线参数 EC 下, 用户随机选择私钥 e , 满足 $1 < e < N$, 生成公钥 $d = e^{-1} \bmod N$ 。对于椭圆曲线上的点 P_M , 加密 $C = e \cdot P_M$, 解密 $P_M' = d \cdot C = d \cdot (e \cdot P_M) = P_M$ 。记加密双方为 Alice 和 Bob, 他们各自持有一对公私钥 (e_a, d_a) 和 (e_b, d_b) 。Alice 首先对椭圆曲线上的点 P_M 加密得到 $C_0 = e_a \cdot P_M$, 然后 Bob 再对 Alice 的加密结果进行二次加密得到最终密文 $C = e_b \cdot C_0 = e_b \cdot (e_a \cdot P_M)$, 这要与一次加密等效, 即 $C = e \cdot P_M$ 。联立两式, 可以得到:

$$e_b \times e_a = e \bmod N \quad (6)$$

也就是说, 对于选定的椭圆曲线参数 EC , 如果连续两次加密的加密密钥满足式(6), 则可以认为这等效于使用等效加密密钥 e 进行一次 ECC 加密。换句话说, 如果选定了某个等效加密密钥 e , 那么 Alice 可以自由选择一个加密密钥 e_a , 而 Bob 始终可以获得相应的密钥 $e_b = e \times e_a^{-1} \bmod N$, 以确保两次加密后的结果保持不变。解密具有相同的性质, 不再赘述。

SECC-DOAEP++ 算法如图 2 所示, 包含 3 个伪随机序列发生器 G 、 H_1 、 H_2 , 一条椭圆曲线 EC , 两个加密密钥 e_a 和 e_b , 两个解密密钥 d_a 和 d_b , 一个等效加密密钥 e 和等效解密密钥 d 。具体步骤如下。

1) 选择一条确定的椭圆曲线, 获得其参数 $EC = (p, a, b, N)$ 。

2) 选择一个等效加密密钥 e , 满足 $1 < e < N$, 并计算等效解密密钥 $d = e^{-1} \bmod N$ 。

3) 随机选择加密密钥 e_a 和解密密钥 d_a , 同样满足 $1 < e_a/d_a < N$, 并计算对应的加密密钥 $e_b = e \times e_a^{-1} \bmod N$, 解密密钥 $d_b = d \times d_a^{-1} \bmod N$ 。

4) 生成 3 个伪随机序列发生器 G 、 H_1 、 H_2 , 其实际输出长度由消息 m 决定。

5) 使用 3 个伪随机序列发生器 G 、 H_1 、 H_2 填充消息 m , 得到 $M = y_3 \| y_4 = y_1 \| y_2$, $y_1 = (m \| \text{Const}) \oplus H_1(G(m))$, $y_2 = G(m) \oplus H_2(y_1)$ 。先将 y_3 编码成曲线上的点 P_{y_3} , 然后使用加密密钥 e_a 和 e_b 先后对 P_{y_3} 进行加密, 得到密文 $C = P_{y_3} \| y_4 = e_b \cdot (e_a \cdot P_{y_3}) \| y_4$, 该密文等效于一次加密 $C = P_{y_3} \| y_4 = e \cdot P_{y_3} \| y_4$ 。

6) 使用解密密钥 d_a 和 d_b 先后对密文 C 中的 P_{y_3} 进行部分解密并解码, 得到填充结果 $M = y_1 \| y_2 = y_3' \| y_4 = d_a \cdot (d_b \cdot P_{y_3}) \| y_4$, 等效于一次解密 $M = y_1 \| y_2 = y_3' \| y_4 = d \cdot P_{y_3} \| y_4$ 。然后使用伪随机序列发生器 H_1 、 H_2 从填充结果 M 中恢复出消息 m , $m \| \text{Const} = y_1 \oplus H_1(y_2 \oplus H_2(y_1))$ 。

这种分步加密后等效的性质在加密去重中非常有意义。如果使用一个可信的密钥生成中心为相同的文件生成唯一的等效加密密钥, 那么无论用户选择使用某个加解密密钥, 密钥生成中心总

能生成一个重加密密钥（和重解密密钥）给云服务器，使文件经云服务器重加密后保持相同。在解密时，经过云服务器的预解密后，用户总能用自己设定的解密密钥恢复文件。显然，云服务器无法从密文中获得任何有效信息。

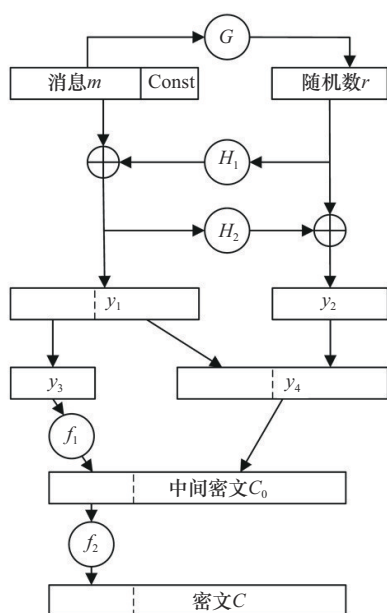


图2 SECC-DOAEP++算法

Figure 2 SECC-DOAEP++ algorithm

2 系统概述

本节将介绍系统模型和威胁模型。表1给出了一些符号描述。

自定义密钥的加密去重云存储访问控制方案中系统包含5个实体，分别是数据所有者、云服务器、密钥生成中心、属性权威和数据使用者。它们之间的关系和信息流转如图3所示。用不同颜色表示的数据所有者、数据使用者和属性权威分别属于不同的属性群，它们由各自的属性权威单独初始化和维护访问控制部分。但是只有唯一的云服务器和密钥生成中心。图中的括号表示访问控制部分，这部分是可选的。

(1) 数据所有者

为了减少存储压力，方便文件共享，数据所有者将文件外包给云服务器。如果不希望共享文件，它们则选择自己的密钥并借助密钥生成中心对文件进行加密上传。如果希望符合访问策略的用户都可以下载并解密文件，它们则借助属性权威对解密

密钥进行属性加密，并上传该文件密钥。

表1 符号描述

Table 1 Notations and descriptions

符号	说明
F	用户待上传文件
FID	文件唯一标识符
u	用户唯一标识符
C	云存储的最终密文
C_0/C'_0	中间加密密文/中间解密密文
K	文件密钥
EC	椭圆曲线参数 $EC = (p, a, b, N)$
K_e/K_d	等效加密密钥和等效解密密钥
K_{eu}/K_{du}	用户端加密密钥和用户端解密密钥
K_{ec}/K_{dc}	云端加密密钥和云端解密密钥
K_A	属性私钥
KGC- T	KGC维护表 $\{FID, K_e, K_d, \{u\}\}$
CSS- T	CSS维护表 $\{FID, ptr, \{u, K, K_{dc}\}\}$

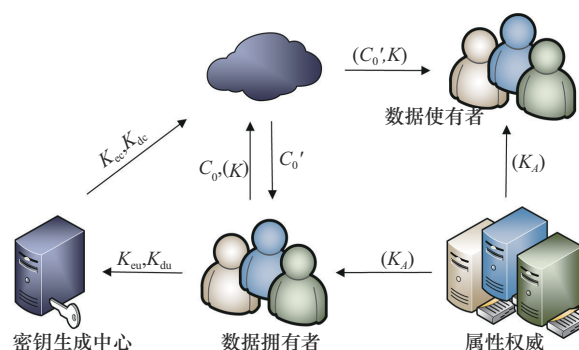


图3 系统模型

Figure 3 System model

(2) 云服务器

云服务器提供海量的存储空间和管理系统。它存储用户上传的文件和文件密钥，直到数据所有者删除它们。云服务器还维护云服务器维护表 CSS- T ，以便能够快速定位存储的文件及其对应的文件密钥。

(3) 密钥生成中心

密钥生成中心为每个相同的文件生成相同的（等效）加密/解密密钥，更新并维护密钥生成中心维护表 KGC- T 。然后根据用户端加解密密钥计算并分发云端加解密密钥给云服务器。

(4) 属性权威

属性权威初始化访问控制方案，将生成的属性

私钥分发给数据拥有者和数据使用者，并负责维护访问控制系统，比如执行属性撤销相关操作。

(5) 数据使用者

数据使用者是从属性权威获得属性私钥的合法用户。根据文件标识和数据拥有者标识，它们下载并尝试解密从云服务器接收到的密文。只有符合访问策略的数据使用者才能正确解密密文。

威胁模型假设云服务器是半可信的，诚实且好奇，云服务器会正确执行系统的指令，但同时尝试从密文中获取有效信息。数据拥有者、密钥生成中心和属性权威是可信的。数据使用者和敌手是不可信的，它们总是试图恢复明文，不管它们是否有权限。并且假设文中所有的通信过程都是安全可靠的。因此，该模型可能面临以下安全威胁：①云服务器和敌手共谋或单独地从所有公开信息中恢复明文；②云服务器和数据使用者共谋解密超出该数据使用者访问权限的密文。

3 方案细节

本文提出的自定义密钥的加密去重云存储访问控制方案同时满足加密重复数据删除和安全文件共享的要求，同时允许每个用户自定义其密钥。本节将介绍整个包括访问控制的方案，只在数据拥有者解密这一小节中对没有访问控制的数据拥有者解密进行简要的介绍。数据拥有者使用任意设置的客户端加密密钥 K_{cu} 对文件 F 进行加密，并使用基于属性的加密对解密密钥 K_{du} ($K_{du} = K_{cu}$)进行加密，然后将中间加密密文 C_0 和文件密钥 K 上传到云服务器。云服务器使用来自密钥生成中心的云端加密密钥 K_{cc} 对中间加密密文 C_0 进行重加密，得到最终存储的密文 C 。当数据使用者请求下载文件 F 时，云服务器首先使用云端解密密钥 K_{dc} 对存储密文 C 进行解密，并将中间解密密文 C_0' 和文件密钥 K 返回给数据使用者。数据使用者使用来自属性权威的属性私钥 K_A 解密文件密钥 K ，然后使用解密结果解密中间解密密文 C_0' 。当且仅当数据使用者的属性满足访问策略时，它才能解密文件密钥 K 并恢复文件 F 。

本文专注于加密去重方案的系统架构，访问控制方案则选取通用的密文策略的属性基加密模型进行说明。在属性基加密的具体实现上，采用与文献[21]相同的方法。它包含4个算法，即初

始化、密钥生成、加密和解密。为了区分，本文称之为属性初始化、属性私钥生成、属性加密以及属性解密。

3.1 系统初始化

系统初始化包含3个算法，分别是属性初始化 ASetup，属性私钥生成 AKeygen 和去重初始化 DSetup，其形式化定义如下。

(1) ASetup(λ, T) $\rightarrow$ (PK, MSK)

属性初始化算法由属性权威执行，输入系统安全参数 λ 和包含所有用户标识的二进制树 T ，输出系统主密钥 MSK 和系统公开参数 PK。属性权威秘密保存 MSK，并发布系统公共参数 PK。

(2) AKeygen (MSK, S, u) $\rightarrow K_A$

属性私钥生成算法由属性权威执行，输入系统主密钥 MSK，属性集合 S 和用户标识 u ，生成用户 u 的属性私钥 K_A ，并将其分发给对应的用户 u 。

(3) DSetup(η) $\rightarrow$ (EC, H, H_f, G, H_1, H_2)

去重初始化算法由密钥生成中心执行，输入一个安全参数 η ，输出椭圆曲线参数 $EC = (p, a, b, N)$ ，一个将口令映射成密钥的哈希函数 H ，一个生成文件唯一标识符的抗碰撞哈希函数 H_f ，3个用于最优非对称加密填充的伪随机序列发生器 G, H_1, H_2 。公开 EC, H, H_f, G, H_1, H_2 。

属性初始化和属性私钥生成算法由属性权威执行。它初始化属性群和云服务器，生成并分发属性私钥给属性群中的每个用户（数据拥有者和数据使用者）。具体细节参照文献[21]，本文只提供形式化定义和描述。

去重初始化算法由密钥生成中心执行，输入一个安全参数 η ，生成一条选定的椭圆曲线，两个哈希函数和3个伪随机序列发生器。首先，密钥生成中心根据安全参数 η ，选定一条具体的椭圆曲线。安全参数 η 决定了椭圆曲线的位数。然后，生成一个哈希函数 H 将口令转换成可用于加密的客户端加密密钥 K_{cu} 。对于素数阶 N 的椭圆曲线，合法的加密密钥只要取任意大于1小于 N 的整数即可，所以哈希函数 H 只需要将任意比特的字符串映射成 $[N]$ 位的01串即可。接着，选择一个抗碰撞的哈希函数 H_f （例如 SHA-512），该函数用于生成文件的唯一标识符 FID。最后，规定3个用于填充的伪随机序列发生器 $G, H_1,$

H_2 , 伪随机序列发生的输入输出长度主要由消息长度决定。密钥生成中心将 EC 、 H 、 H_f 、 G 、 H_1 、 H_2 全部公开。用 $[x]$ 表示参数 x 的比特长度, 那么可以将结果用以下式子表示:

$$[N] = \eta \quad (7)$$

$$H: \{0,1\}^* \rightarrow \{0,1\}^{[N]} \quad (8)$$

$$H_f: \{0,1\}^* \rightarrow \{0,1\}^{[FID]} \quad (9)$$

$$G: \{0,1\}^{[m]+[Const]} \rightarrow \{0,1\}^{[r]} \quad (10)$$

$$H_1: \{0,1\}^{[r]} \rightarrow \{0,1\}^{[m]+[Const]} \quad (11)$$

$$H_2: \{0,1\}^{[m]+[Const]} \rightarrow \{0,1\}^{[r]} \quad (12)$$

其中, $\{0,1\}^*$ 表示任意长度的01比特串。

3.2 加密上传

加密上传过程主要包含4个算法, 分别是ECC等效密钥生成DKeygen, 用户端ECC加密DEncryptU, 属性加密AEncrypt和云端ECC加密DEncryptC, 其形式化定义如下。

1) DKeygen(EC, FID) $\rightarrow (K_e, K_d)$

ECC等效密钥生成算法由密钥生成中心执行, 输入椭圆曲线 EC 和文件标识符 FID , 输出ECC等效加密密钥 K_e 和等效解密密钥 K_d , 并记录在 $KGC-T = \{FID, K_e, K_d, \{u\}\}$ 中。

2) DEncryptU($EC, K_{eu}, G, H_1, H_2, F$) $\rightarrow C_0$

用户端ECC加密算法是使用DOAEP++的ECC加密算法, 由数据拥有者执行, 输入椭圆曲线参数 EC , 用户端加密密钥 K_{eu} , 3个伪随机序列发生器 G 、 H_1 、 H_2 和明文 F , 输出中间加密密文 C_0 , 并将中间加密密文发送给云服务器。

3) AEncrypt(PK, W, K_{du}) $\rightarrow K$

属性加密算法由数据拥有者执行, 输入系统公共参数 PK , 访问策略 W 和用户端解密密钥 K_{du} , 输出文件密钥 K , 并将文件密钥发送给云服务器。

4) DEncryptC(EC, K_{ec}, C_0) $\rightarrow C$

云端ECC加密算法是一般的ECC加密算法, 由云服务器执行, 输入椭圆曲线参数 EC , 云端加密密钥 K_{ec} 和中间加密密文 C_0 , 生成最终云服务器存储的密文 C 。

文件的加密上传过程涉及数据拥有者、密钥生成中心和云服务器三者的通信过程, 假设它们之间已经建立好一个安全的信道, 加密上传流程如图4所示。

首先, 数据拥有者使用哈希函数 H_f 计算待

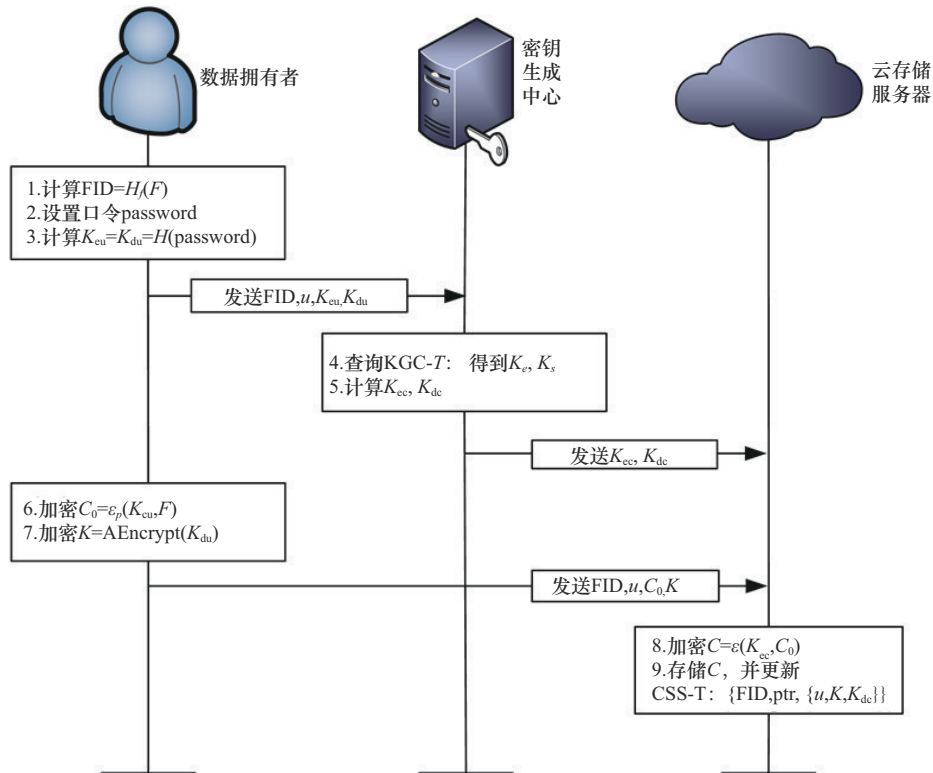


图4 加密上传流程

Figure 4 The procedure of encryption and upload

上传文件的唯一文件标识符 FID，并设定一个加解密口令 password，计算加解密密钥 K_{cu} 、 K_{du} 。口令和哈希函数的组合主要是为了用户能够自由地设置它们喜欢的口令，而无须记忆冗长的椭圆曲线密钥，并且这样的处理更加符合实际的应用场景。然后数据拥有者将文件标识符 FID，自己的用户标识 u 和加解密密钥 K_{cu} 、 K_{du} 发送给密钥生成中心。具体的计算过程如下所示：

$$FID = H_f(F) \quad (13)$$

$$K_{cu} = K_{du} = H(\text{password}) \quad (14)$$

密钥生成中心收到文件标识符 FID，用户标识 u 和加解密密钥 K_{cu} 、 K_{du} 之后，首先查找 KGC- T 表中是否存在 FID 的表项。如果不存在，则说明该数据拥有者 u 是该文件 F 的首个上传者，密钥生成中心需要根据椭圆曲线参数 EC 和文件标识 FID 执行 ECC 等效密钥生成算法，生成文件的等效加密密钥 K_e 和等效解密密钥 K_d ，并将表项 $\{FID, K_e, K_d, \{u\}\}$ 添加到 KGC- T 中。否则，数据拥有者是文件 F 的后续上传者，密钥生成中心直接从表项中获得等效加密密钥 K_e 和等效解密密钥 K_d 。然后，密钥生成中心计算 K_{cc} 和 K_{dc} ，并将 K_{cc} 、 K_{dc} 发送给云服务器。具体的计算过程如下所示：

$$K_{eR} \xleftarrow{FID} (1, N) \quad (15)$$

$$K_d = K_e^{-1} \bmod N \quad (16)$$

$$K_{cc} = K_e \times K_{cu}^{-1} \bmod N \quad (17)$$

$$K_{dc} = K_d \times K_{du}^{-1} \bmod N \quad (18)$$

然后数据拥有者可以使用用户端 ECC 加密算法加密待上传文件 F 和属性加密算法加密用户端解密密钥 K_{du} 。首先使用 3 个伪随机序列发生器 G 、 H_1 、 H_2 对待上传文件 F 进行 DOAEP++，得到填充结果 $M = y_3 || y_4 = y_1 || y_2, y_1 = (m || \text{Const}) \oplus H_1(G(m)), y_2 = G(m) \oplus H_2(y_1)$ 。然后使用 Koblitz 标准方法^[22]对 y_3 进行编码，其中 l 为保留用于填充的固定长度。

算法 1 ECC 编码方式

输入 $\eta - 1 - l$ 比特的消息 m ，位长为 η 比特的椭圆曲线参数 $EC = (p, a, b, N)$

输出 一个椭圆曲线上的点 (x, y)

- 1) $x = m || 0^l$
- 2) 将 x 从字节类型转换成数
- 3) $yy = x^3 + ax + b \bmod p$
- 4) **while** yy 不是一个二次剩余 **do**
- 5) $x = x + 1$
- 6) $yy = x^3 + ax + b \bmod p$
- 7) **end while**
- 8) 计算 $y = \sqrt{yy} \bmod p$ ，得到其中较小的解
- 9) 返回 (x, y)

该编码方式的思想是将消息转换成椭圆曲线上点的横坐标 x ，并计算相应的纵坐标 y 。但由于椭圆曲线上的点不是连续的，直接转换得到的横坐标值并不一定在曲线上。所以需要减少消息的比特长度，将最后几位长度保留用于填充（该长度即为 l ），直到找到一个合适的填充使 x 坐标成为曲线上合法点的横坐标。如果保留的位数取遍所有的值也无法满足则编码失败。否则计算此时相应的 y 值，这时的点 (x, y) 是满足椭圆曲线的一个点。然后计算编码失败的概率。由于素数 p 的二次剩余集合大小为 $(p - 1)/2$ ，所以任意一个 x 计算得到的 $yy = x^3 + ax + b \bmod p$ 是二次剩余的概率约为 $1/2$ 。那么该算法失败的概率约为 2^{-l} 。只需要取合适的 l ，失败的概率可以忽略不计。编码完成后，数据拥有者使用用户端加密密钥 K_{cu} 对编码结果 $P_{y_3}(x, y)$ 进行 ECC 加密，得到中间加密密文 C_0 。最后，数据拥有者将 FID、 u 、 C_0 和 K 发送给云服务器。中间加密密文和文件密钥由以下式子得到，其中 $\mathcal{E}_p$ 表示带 DOAEP++ 的 ECC 加密。

$$C_0 = \mathcal{E}_p(EC, K_{cu}, G, H_1, H_2, F) \quad (19)$$

$$K = \text{AEncrypt}(PK, W, K_{du}) \quad (20)$$

当云服务器收到用户发来的文件唯一标识符 FID，用户标识 u ，用户加密后的中间加密密文 C_0 和文件密钥 K ，以及密钥生成中心发来的椭圆曲线参数 EC，云端加密密钥 K_{cc} 和云端解密密钥 K_{dc} 后，执行 DEncryptC 算法用云端加密密钥对中间加密密文 C_0 进行二次加密，生成云服务器最终存储的密文 C 。然后云服务器更新其数据表 $CSS - T = \{FID, ptr, \{u, K, K_{dc}\}\}$ 。ptr 是一个文件指针，指向存储文件的具体位置，用于云服务

器快速定位文件。云服务器存储的密文由式 (21) 得到：

$$C = \mathcal{E}(\text{EC}, K_{\text{ec}}, C_0) \quad (21)$$

其中, $\mathcal{E}$ 表示一般的 ECC 加密。

3.3 下载解密

下载解密过程主要包含 3 个算法, 分别是云端 ECC 解密 DDecryptC、属性解密 ADecrypt 和客户端 ECC 解密 DDecryptU, 其形式化定义如下。

$$1) \text{DDecryptC}(\text{EC}, K_{\text{dc}}, C) \rightarrow C_0'$$

云端 ECC 解密算法是一般的 ECC 解密算法, 由云服务器执行。输入椭圆曲线参数 EC, 云端解密密钥 K_{dc} 和云存储的最终密文 C , 生成中间解密密文 C_0' 。并将中间解密密文返回给请求下载的用户。

$$2) \text{ADecrypt}(K_A, K) \rightarrow K_{\text{du}}$$

属性解密算法由数据使用者执行, 输入数据使用者的属性私钥 K_A 和文件密钥 K , 解密得到用户端解密密钥 K_{du} 。

$$3) \text{DDecryptU}(\text{EC}, K_{\text{du}}, G, H_1, H_2, C_0') \rightarrow F$$

用户端 ECC 解密算法是使用 DOAEP++ 的 ECC 解密算法, 由数据使用者执行, 输入椭圆曲线参数 EC, 用户端解密密钥 K_{du} , 3 个伪随机序列发生器 G, H_1, H_2 和中间解密密文 C_0' , 解密得到原始明文 F 。

数据使用者下载解密过程同样涉及数据使用者、密钥生成中心和云服务器三者的通信过程。数据使用者使用文件唯一标识符 FID 和数据拥有者的用户标识 u 从云服务器下载经云服务器解密后的中间解密密文 C_0' 和文件密钥 K 。然后, 数据使用者先使用自己的属性私钥 K_A 对文件密钥 K 进行解密, 再用解密结果对中间解密密文 C_0' 进行解密。如果数据使用者的属性满足数据拥有者设定的访问策略, 那么数据使用者就可以成功解密并恢复原始明文 F 。

当云服务器接收到数据使用者发来的包含文件唯一标识符 FID 和用户标识 u 的下载请求时, 云服务器从表 $\text{CSS} - T$ 中找到对应的文件路径 ptr, 文件密钥 K 和云端解密密钥 K_{dc} 。云服务器从文件路径 ptr 中取出密文 C 后, 先使用云端解密密钥 K_{dc} 对密文 C 进行 ECC 解密, 得到中间解密密文 C_0' 。然后, 云服务器将中间解密密文 C_0'

和文件密钥 K 发送给数据使用者。用 $\mathcal{D}$ 表示 ECC 解密, 中间解密密文可以由式 (22) 得到:

$$C_0' = \mathcal{D}(\text{EC}, K_{\text{dc}}, C) \quad (22)$$

当数据拥有者收到中间解密密文 C_0' 和文件密钥 K 后, 可以根据自己的属性私钥 K_A 独立地完成解密工作, 恢复出原始明文。首先, 数据拥有者使用属性私钥 K_A 调用属性解密算法对文件密钥 K 进行解密。当且仅当数据使用者的属性满足设定的访问策略并且不在撤销列表中时, 数据使用者可以成功解密得到用户端解密密钥 K_{du} 。然后, 数据使用者使用用户端解密密钥 K_{du} 对中间解密密文 C_0' 进行解密和去填充, 恢复出原始明文 F 。用 $\mathcal{D}_p$ 表示带 DOAEP++ 的 ECC 解密算法, 数据使用者的解密过程可以用式 (23)、式 (24) 表示:

$$K_{\text{du}} = \text{ADecrypt}(K_A, K) \quad (23)$$

$$F = \mathcal{D}_p(\text{EC}, K_{\text{du}}, G, H_1, H_2, C_0') \quad (24)$$

3.4 数据拥有者解密

本文还为数据拥有者提供了其他的解密方式。数据拥有者可以使用自己设定的口令, 在不通过上述属性解密的情况下解密密文。这意味着不希望共享文件的用户可以使用它们自己设定的密码来加密和解密它们的文件, 与此同时, 云服务器仍然能够将这些用户上传的密文与其他用户上传的密文一起执行去重操作。这使用户可以根据自己的需求自由地选择是否需要采取额外的措施 (增设属性权威维护属性群) 来实现安全的文件共享, 同时不会增加云服务器的负担。这为当前云存储中, 用户想要加解密自己的文件, 并且只在需要分享时采取额外的访问控制策略这一实际需求 and 云服务器的去重要求找到了一个平衡。

数据拥有者的下载过程与数据使用者完全相同, 只是解密过程有所区别。数据拥有者可以使用口令 password 解密从云服务器下载的中间解密密文 C_0' , 这是因为在设置分步 ECC 的密钥时, 已经约定了 $K_{\text{du}} = K_{\text{cu}} = H(\text{password})$ 。这使方案虽然使用的是 ECC 非对称加密, 但对用户来说, 体验与对称加密没有区别, 它们使用同一个密钥进行加解密。之后, 数据拥有者使用 K_{du} 对中间解密密文 C_0' 进行解密。数据拥有者的解密过程

可以用式 (25)、式 (26) 表示:

$$K_{du} = H(\text{password}) \quad (25)$$

$$F = \mathcal{D}_p(\text{EC}, K_{du}, G, H_1, H_2, C_0') \quad (26)$$

4 方案分析与比较

4.1 安全性分析

自定义密钥的加密去重云存储访问控制方案主要包含两个算法, SECC-DOAEP++ 和 CP-ABE。CP-ABE 只是用于加解密用户端的解密密钥 K_{du} , 只有当数据使用者的属性满足设定的访问策略并且数据使用者不在撤销列表中的时候, 数据使用者才能正确解密并恢复出用户端解密密钥 K_{du} 。其安全性由具体的密文策略的属性基加密方案确定, 一般可以归约到双线性 Diffie-Hellman 困难性问题上。本方案中的属性基加密使用了文献[21]中介绍的方案, 其已经证明了定理 1, 其中 TRAP-CPABE 是文献[21]中属性基加密方案的名称。

定理 1 如果 q -BDHE 困难性假设成立, 那么没有一个多项式敌手能够在选择访问策略和选择明文攻击下以不可忽略的优势攻破 TRAP-CPABE 方案, 满足 $q > 2|U| - 2$, $|U|$ 是系统中用户的数量。

SECC-DOAEP++ 可以分以下两个步骤进行分析: ① ECC-DOAEP++ 满足 D-INDS-CPA 安全性; ② SECC 的安全性可以归约到 ECC 问题上。其中, D-INDS 即确定性密文不可区分性, 与一般的密文不可区分性^[23]相似, 只是敌手被严格要求只能使用不同的消息访问加密谕言机^[20]。并且 D-INDS-CPA 安全性是当前加密去重范式下所能达到的最高安全性^[20]。

定理 2 假设存在一个 D-INDS-CPA 敌手 $\mathcal{A}$ 以 ε 的优势解决了 ECC-DOAEP++ 问题, 那么一定存在一个敌手 $\mathcal{B}$ 能够以 ε' 的优势解决椭圆曲线上的离散对数问题 (elliptic curves discrete logarithm problem, ECDLP), 用 q_{H_1} 表示敌手 $\mathcal{A}$ 询问随机谕言机 H_1 的次数, 则有 $\varepsilon' \geq \varepsilon - q_{H_1}/2^{\lceil r \rceil}$ 。

证明 首先, 给出攻击 ECC-DOAEP++ 的 D-INDS-CPA 游戏:

$$\text{Exp}_A^{\text{ECC-DOAEP++}}(\eta):$$

$$(f, g) \leftarrow \text{Gen}(\eta);$$

$$G \leftarrow_R \{ G: \{0,1\}^{\lceil y_1 \rceil} \rightarrow \{0,1\}^{\lceil r \rceil};$$

$$H_1 \leftarrow_R \{ H_1: \{0,1\}^{\lceil r \rceil} \rightarrow \{0,1\}^{\lceil y_1 \rceil};$$

$$H_2 \leftarrow_R \{ H_2: \{0,1\}^{\lceil y_1 \rceil} \rightarrow \{0,1\}^{\lceil r \rceil};$$

$$(m_0, m_1) \leftarrow \mathcal{A}^{G, H_1, H_2}(f);$$

$$\beta \leftarrow_R \{0,1\}; C = f(y_3) \parallel y_4,$$

$$y_3 \parallel y_4 = y_1 \parallel y_2,$$

$$y_1 = (m_\beta \parallel \text{Const}) \oplus H_1(r),$$

$$y_2 = r \oplus H_2(s), r = G(m_\beta),$$

$$[y_3] = k_1, [y_1] = [m_\beta \parallel \text{Const}], [y_2] = [r];$$

$$\beta' \leftarrow \mathcal{A}(f, c);$$

如果 $\beta' = \beta$, 返回 1; 否则, 返回 0。

然后, 使用文献[19]对其定理 2 的证明思路进行证明。构造敌手 $\mathcal{B}$, $\mathcal{B}$ 首先初始化 3 个空的列表, 分别为 G -list, H_1 -list 和 H_2 -list。这些列表分别用于描述相应伪随机序列发生器的输入和输出。在敌手 $\mathcal{A}$ 的查询阶段, $\mathcal{B}$ 利用这 3 个列表对 $\mathcal{A}$ 的随机谕言机 G 、 H_1 、 H_2 进行如下模拟。当 $\mathcal{A}$ 以 $\hat{g}$ 询问随机谕言机 G 时, $\mathcal{B}$ 随机生成一个 $\hat{G}$ 用来回答 $\mathcal{A}$, 并且将 $\hat{g}$ 和 $\hat{G}$ 加入列表 G -list 中。同样将 $\hat{h}_1$ 和 $\hat{H}_1$ 加入 H_1 -list 中, 将 $\hat{h}_2$ 和 $\hat{H}_2$ 加入 H_2 -list 中。 $\mathcal{A}$ 生成 (m_0, m_1) 作为它要挑战的明文。

令 $y_4 = y_5 \parallel y_2$, 即 $y_1 \parallel y_2 = y_3 \parallel y_5 \parallel y_2$ 。 $\mathcal{B}$ 随机选择则 $\beta \in \{0,1\}$, r 和 $y_3 \parallel y_2$, 然后定义 G 、 H_1 、 H_2 , 生成 m_β (m_β 即为 $\mathcal{B}$ 选取的 m_0 或 m_1) 的密文 $C = \bar{c} \parallel y_4$, $\bar{c}$ 即是 $\mathcal{B}$ 要尝试解决的 ECDLP 问题 (已知 $\bar{c} = f(y_3) = e \cdot P_{y_3}$, 求解 P_{y_3} 或 y_3 , 编解码算法是公开的):

$$G(m_\beta) \triangleq r^* \quad (27)$$

$$H_1(r^*) \triangleq (m_\beta \parallel \text{Const}) \oplus (y_3 \parallel y_5) \quad (28)$$

$$H_2(y_3 \parallel y_5) \triangleq y_2 \oplus r^* \quad (29)$$

$\mathcal{B}$ 同样将 m_β 和 r^* 加入 G -list 中, r^* 和 $(m_\beta \parallel \text{Const}) \oplus (y_3 \parallel y_5)$ 加入 H_1 -list 中, $y_3 \parallel y_5$ 和 $y_2 \oplus r^*$ 加入 H_2 -list 中。然后在敌手 $\mathcal{A}$ 的猜测阶段, 根据定理 2 的假设, 如果 $\mathcal{B}$ 能够正确地模拟 3 个伪随机序列发生器, 则 $\mathcal{A}$ 能以 ε 的优势区分 β 。但是 $\mathcal{B}$ 并不能正确地模拟 m_β 、 r^* 的询问, 因为 $\mathcal{B}$ 不知道 y_3 。首先可以忽略对随机谕言机 G 的询问基于以下两个事实: ① $\hat{G}$ 仅被用来生成 r^* , 却不参与后续的计算, 后续计算使用 $\hat{h}_1$ 的值代替

(这和文献[18]引理2的证明中构造Game2的处理方式相同)；② D-IND\$要求没有任何相同的消息 m 会被询问两次，即 m_β 从来没有被提交给随机谕言机 G 查询。所以敌手 $\mathcal{A}$ 无法区分 $\mathcal{B}$ 对伪随机序列发生器 G 的模拟。至于随机谕言机 H_1 和 H_2 ，考虑以下两种情况。

1) Ask H_2 表示 $(y_3||*)$ 在 q_{H_2} 次对 H_2 的询问中已经被提交询问过，且该询问比 r^* 被提交给 H_1 的时间早，其中 $*$ 表示任意字符。

2) Ask H_1 表示 r^* 在 q_{H_1} 次对 H_1 的询问中已经被提交询问过，且该询问比 $(y_3||*)$ 被提交给 H_2 的时间早。

那么，根据文献[19]中的结论， $\mathcal{B}$ 能够以 β' 的优势从密文 $\bar{c}$ 中恢复出完整的明文 y_3 ， ε' 满足：

$$\varepsilon' = \Pr(\neg \text{Ask}H_1 \wedge \text{Ask}H_2) \geq \varepsilon - q_{H_1}/2^{[r]} \quad (30)$$

定理3 如果存在一个多项式时间的敌手 $\mathcal{A}$ 能够解决SECC问题，那么一定存在一个敌手 $\mathcal{B}$ 能够解决ECC问题。

在形式化证明前，先简单地分析SECC的安全性。容易知道，SECC问题和ECC问题有着相同的数学形式。对于ECC问题，知道加密密钥 e 的敌手无法得到解密密钥 d ，满足 $e \times d = 1 \bmod N$ 。而在本方案的SECC中， $K_{\text{eu}} \times K_{\text{ec}} = K_e \bmod N$ ， $K_{\text{du}} \times K_{\text{dc}} = K_d \bmod N$ ，所以， $(K_{\text{eu}} \times K_{\text{du}}) \times (K_{\text{ec}} \times K_{\text{dc}}) = 1 \bmod N$ 。这在数学形式上与ECC等价，即知道 K_{ec} 、 K_{dc} 的敌手无法知道解密密钥 K_{eu} 、 K_{du} 。并且由于在威胁模型中，数据拥有者和密钥生成中心是可信的， K_{du} 是由CP-ABE方案保护的，所以只需要保证云服务器和外部敌手无法恢复明文。云服务器比外部敌手多知道 K_{ec} 和 K_{dc} ，因此，在下面的证明过程中只考虑了云服务器的攻击视图。

证明 构造敌手 $\mathcal{B}$ 如下。 $\mathcal{B}$ 想要从一组ECC问题 (EC, e, c) 中恢复出明文 m （认为该明文已经被编码成椭圆曲线上的点），其中， EC 为椭圆曲线参数， e 为加密密钥， c 为密文，满足椭圆曲线上的点乘运算 $c = e \times m$ 。 $\mathcal{B}$ 随机选择 r_1 和 r_2 ，并计算 $c_1 = r_1 \cdot c$ ， $c_2 = r_2 \cdot c_1$ 。然后 $\mathcal{B}$ 将 $(EC, e, c, r_1, c_1, r_2, c_2)$ 发送给 $\mathcal{A}$ 。对于敌手 $\mathcal{A}$ ，这恰恰就是其攻击SECC的视图，也是云服务器在该

方案中的视图，即 $(EC, K_{\text{eu}}, C_0, K_{\text{ec}}, C, K_{\text{dc}}, C_0')$ 。因此，如果 $\mathcal{A}$ 能够解决SECC问题，那么 $\mathcal{B}$ 一定可以通过以上构造解决ECC问题。

综上，由定理1~定理3，可以证明该方案的安全性。并且，类似于一次一密加密，该方案中的密钥是与文件绑定的，即一文件一密钥。该方案为每一个不同的文件生成不同的加解密密钥，从而进一步提高方案的安全性。

4.2 功能比较

将本方案与DupLESS^[6]、LEVER^[14]、Yang's^[24]、SED^[15]、Guo's^[16]方案在主要功能和算法上进行比较。表2显示了只有本方案可以支持数据拥有者自定义密钥和支持多属性群同时工作，而在访问控制上，本方案也比其他方案有更高的细粒度。从表3可以看出，大部分方案使用AES对称加密来保证数据的保密性，因为具有更快的加密效率。而本方案使用了SECC-DOAEP++来使用户可以自定义它们的密钥，虽然牺牲了部分性能，但更能满足实际应用场景。在密钥统一算法上，本方案与Yang's和SED有接近的效率，比LEVER快，比DupLESS^[6]和Guo's慢。

表2 主要功能比较

Table 2 Comparison of the main functionality			
方案	自定义密钥	访问控制	多属性群
DupLESS ^[6]	否	否	否
LEVER ^[14]	否	否	否
Yang's ^[24]	否	是(基于ID)	否
SED ^[15]	否	是(基于认证)	否
Guo's ^[16]	否	是(基于ID)	否
本文	是	是(基于属性)	是

4.3 性能分析

本文方案在Ubuntu 20.04 LTS上，基于python的pycryptodome和charm-crypto库实现了代码，实验的机器CPU型号为AMD Ryzen 5 5600 GB with Radeon Graphics 3.90 GHz，基带RAM为8 GB。图5展示了本文方案中主要算法的运行时间。图中DEncryptU和DDecryptU以左侧坐标轴为纵轴，DEncryptC和DDecryptC以右侧坐标轴为纵轴。实验时选择椭圆曲线NIST P-256，测

表 3 主要实现算法比较

Table 3 Comparison of the main algorithms

方案	保密性算法	密钥统一算法	标签一致性算法
DupLESS ^[6]	AES-CTR	带签名的加密	SHA-256
LEVER ^[14]	AES-CBC	同态加密	SHA-512
Yang's ^[24]	AES	双线性对和 BGN 加密	布隆过滤器
SED ^[15]	AES128	双线性对或 RSA	SHA1
Guo's ^[16]	AES	椭圆曲线加密和广播加密	SHA-256
本文	SECC-DOAEP++	密文策略的属性基加密	SHA-512

试文件和用户端加密密钥 K_{eu} 是随机生成的, 其他参数对应生成。加解密时间文件大小变化如图 5 所示, 从图 5 可以看出, 用户端加解密的时间明显大于云端加解密的时间, 这是因为用户端需要对数据进行 DOAEP++ 填充或去填充, 而云端不需要。另外, 用户端加密时间大于用户端解密时间, 这主要是由于加密过程中需要将字符串编码成椭圆曲线上的点 (即算法 1), 这一过程需要对填充值依次尝试, 而解码过程不需要。进一步分析可以知道, 用户端加解密的时间主要消耗在 DOAEP++ 上, 填充过程 (或去填充过程) 中需要使用伪随机序列发生器生成与待加密文件几乎等长的伪随机序列, 并与之逐比特异或。而 DOAEP++ 的引入是本方案达到当前加密去重范式下最高安全性 D-IND $\mathcal{S}$ -CPA 必须的, 所以算法的执行效率难以进一步提升。虽然 SECC-DOAEP++ 算法的运行时间相较一般的对称加密没有明显优势, 但其引入可以使每一个上传文件的用户设定他们自己的加解密密钥, 并且这一速率在日常的上传和下载场景中是可被接受的。

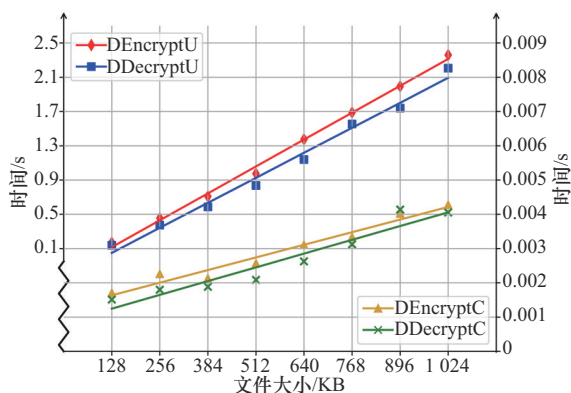


图 5 加解密时间随文件大小变化

Figure 5 Encryption and decryption time for different file sizes

5 结束语

本文构造了一种新的算法 SECC-DOAEP++, 并在此基础上提出了一种自定义密钥的加密去重云存储访问控制方案。本方案使用户能够为他们自己的文件选择自己的加解密密钥, 并可以通过 CP-ABE 方案安全地共享文件。无论用户是否使用 CP-ABE 方案共享他们的文件或使用哪一种 CP-ABE 方案, 云服务器均可以对所有用户上传的加密文件执行重复数据删除操作。安全性证明表明, 在当前的加密去重范式下, SECC-DOAEP++ 达到了最高的安全性 D-IND $\mathcal{S}$ -CPA。在未来, 可以尝试通过边加密边上传的方式进一步提高性能。

参考文献:

- [1] CHRISTEN P. A survey of indexing techniques for scalable record linkage and deduplication[J]. IEEE Transactions on Knowledge and Data Engineering, 2012, 24(9): 1537-1555.
- [2] MEYER D T, BOLOSKEY W J. A study of practical deduplication[J]. ACM Transactions on Storage, 2012, 7(4): 14.
- [3] ALZAIN M A, PARDEDE E, SOH B, et al. Cloud computing security: from single to multi-clouds[C]//Proceedings of the 2012 45th Hawaii International Conference on Systems Science. Piscataway: IEEE Press, 2012: 5490-5499.
- [4] DOUCEUR J R, ADYA A, BOLOSKEY W J, et al. Reclaiming space from duplicate files in a serverless distributed file system[C]//Proceedings of the 22nd International Conference on Distributed Computing Systems. Piscataway: IEEE Press, 2002: 617-624.
- [5] BELLARE M, KEELVEEDHI S, RISTENPART T. Message-locked encryption and secure deduplication[J]. IACR Cryptology EPrint Archive, 2012: 631.
- [6] BELLARE M, KEELVEEDHI S, RISTENPART T. DupLESS: server-aided encryption for deduplicated storage[C]. Proceedings of the 22nd USENIX Conference on Security. New York: ACM, 2013: 179-194.
- [7] LIU J, ASOKAN N, PINKAS B. Secure deduplication of encrypted

- data without additional independent servers[J]. Proceedings of the ACM Conference on Computer and Communications Security, 2015: 874-885.
- [8] YAN Z, WANG M, LI Y, et al. Encrypted data management with deduplication in cloud computing[J]. IEEE Cloud Computing, 2016, 3(2): 28-35.
- [9] XIONG J, ZHANG Y, LI X, et al. RSE-PoW: a role symmetric encryption pow scheme with authorized deduplication for multimedia data[J]. Mobile Networks and Applications, 2018, 23(3): 650-663.
- [10] KWON H, HAHN C, KIM D, et al. Secure deduplication for multimedia data with user revocation in cloud storage[J]. Multimedia Tools and Applications, 2017, 76(4): 5889-5903.
- [11] SONG M, HUA Z, ZHENG Y, et al. FCDedup: A two-level deduplication system for encrypted data in fog computing[J]. IEEE Transactions on Parallel and Distributed Systems, 2023, 34(10): 2642-2656.
- [12] YUAN H, CHEN X, LI J, et al. Secure cloud data deduplication with efficient reencryption[J]. IEEE Transactions on Services Computing, 2022, 15(1): 442-456.
- [13] JIANG T, CHEN X, WU Q, et al. Secure and efficient cloud data deduplication with randomized tag[J]. IEEE Transactions on Information Forensics and Security, 2017, 12(3): 532-543.
- [14] POORANIAN Z, SHOJAFAR M, GARG S, et al. LEVER: secure deduplicated cloud storage with encrypted two-party interactions in cyber-physical systems[J]. IEEE Transactions on Industrial Informatics, 2021, 17(8): 5759-5768.
- [15] ZHANG D, LE J, MU N, et al. Secure and efficient data deduplication in JointCloud storage[J]. IEEE Transactions on Cloud Computing, 2023, 11(1): 156-167.
- [16] GUO C, WANG L, TANG X, et al. Two-party interactive secure deduplication with efficient data ownership management in cloud storage[J]. Journal of Information Security and Applications, 2023, 73: 103426.
- [17] BELLARE M, ROGAWAY P. Optimal asymmetric encryption[C]// Proceedings of Advances in Cryptology—EUROCRYPT'94: Workshop on the Theory and Application of Cryptographic Techniques. Italy: Springer, 1994: 92-111.
- [18] FUJISAKI E, OKAMOTO T, POINTCHEVAL D, et al. RSA-OAEP is secure under the RSA assumption[J]. Journal of Cryptology, 2004, 17(2): 81-104.
- [19] KOBARA K, IMAI H. OAEP++ : a very simple way to apply OAEP to deterministic OW-CPA primitives[J]. IACR Cryptology EPrint Archive, 2002: 130.
- [20] DUAN Y. Distributed key generation for encrypted deduplication: Achieving the strongest privacy[C]// Proceedings of the 6th Edition of the ACM Workshop on Cloud Computing Security, New York: ACM, 2014: 57-68.
- [21] HAN D, PAN N, LI K. A traceable and revocable ciphertext-policy attribute-based encryption scheme based on privacy protection[J]. IEEE Transactions on Dependable and Secure Computing, 2022, 19(1): 316-327.
- [22] KOBLITZ N. Elliptic curve cryptosystems[J]. Mathematics of computation, 1987, 48(177): 203-209.
- [23] ROGAWAY P, BELLARE M, BLACK J. OCB: A block-cipher mode of operation for efficient authenticated encryption[J]. ACM Transactions on Information and System Security, 2003, 6(3): 365-403.
- [24] YANG X, LU R, SHAO J, et al. Achieving efficient secure deduplication with user-defined access control in cloud[J]. IEEE Transactions on Dependable Secure Computing, 2022, 19(1): 591-606.

[作者简介]



谷博伦（1999-），男，浙江温州人，中国科学技术大学硕士生，主要研究方向为数据安全。



徐子凯（1999-），男，山东聊城人，中国科学技术大学博士生，主要研究方向为数据安全。



李卫海（1975-），男，辽宁大连人，中国科学技术大学副教授，主要研究方向为多媒体内容安全、数据安全。



俞能海（1964-），男，安徽无为，人，中国科学技术大学教授、博士生导师，主要研究方向为视频处理与多媒体通信、信息检索、媒体内容安全、数据安全。