基于双层加密和密钥共享的云数据去重方法

高文静[1),2)] 咸鹤群[1),2)] 程润辉[3)]

[1)](青岛大学计算机科学技术学院 山东 青岛 266071)

[2)](中国科学院信息工程研究所信息安全国家重点实验室 北京 100093)

[3)](新华社手机电视台 北京 100053)

摘 要 数据去重技术在云存储系统中的广泛应用，可以有效地节省网络通信带宽，提高云服务器的存储效率. 随着信息安全问题的日益凸显，用户对于数据隐私的重视程度越来越高. 为保护数据隐私，用户普遍将数据加密后上传至云服务器. 相同的数据经过不同用户加密后得到不同的密文，使得云服务器难以进行数据重复性检测. 如何在保护数据隐私的前提下，实现云存储中加密数据的去重，成为研究的热点问题. 现有方案大多借助可信第三方实现云数据安全去重，但可信第三方在现实应用中极难部署，且易成为系统瓶颈. 提出一种基于双层加密和密钥共享的云数据去重方案，无需可信第三方参与，实现云存储中加密数据的安全去重. 通过划分数据流行度，对隐私程度较高的非流行数据采用双层加密机制进行保护. 内层为收敛加密，外层为对称加密. 借助门限秘密共享机制，将外层加密使用的加密密钥保存到多个密钥管理服务器，实现不同用户间的密钥共享. 对隐私程度不高的流行数据，采用简单高效的收敛加密. 安全性分析与性能对比体现本文的方案具有较高的安全性与执行效率. 通过仿真实验，验证了方案的可行性和高效性.

关键词 加密数据去重；双层加密；密钥共享；数据流行度；云存储安全

中图法分类号 TP309 **DOI 号** 10.11897/SP.J.1016.2021.02203

A Cloud Data Deduplication Method Based on Double-Layered Encryption and Key Sharing

GAO Wen-Jing[1),2)] XIAN He-Qun[1),2)] CHENG Run-Hui[3)]

[1)](*College of Computer Science and Technology, Qingdao University, Qingdao, Shandong* 266071)

[2)](*State Key Laboratory of Information Security* (*Institute of Information Engineering, Chinese Academy of Sciences*), *Beijing* 100093)

[3)](*Mobile Television XinHua News Agency, Beijing* 100053)

Abstract Data deduplication is a technique for eliminating duplicate copies of data. It has been widely used in cloud storage systems, which can effectively reserve network communication bandwidth and improve cloud storage efficiency. As information security issues become more severe, users are paying more and more attention to data privacy, they tend to encrypt data before uploading them to the cloud server. Identical data may be encrypted into different ciphertext by different users, which makes it difficult for cloud servers to detect duplication. How to realize the deduplication of encrypted data has become a booming research area in cloud security. Convergent encryption is an effective measure to balance data encryption and data deduplication, and has been widely applied. Some improved data deduplication schemes are proposed on the basis of convergent encryption. However, most existing schemes rely on trusted third parties to ensure the security of data deduplication. User ownership of data depends on

收稿日期：2019-11-11；在线发布日期：2020-08-31. 本课题得到山东省自然科学基金（ZR2019MF058）、国家自然科学基金（61702294）、信息安全国家重点实验室开放课题（2020-MS-09）资助. **高文静**，博士研究生，主要研究领域为密码学、云安全. E-mail: gaowj87@126.com. **咸鹤群**（**通信作者**），博士，副教授，中国计算机学会(CCF)高级会员（29052M），主要研究领域为密码学、云安全、网络安全. E-mail: xianhq@126.com. **程润辉**，硕士，主要研究领域为信息安全.

encryption keys, and there are certain limitations in data availability. Once the encryption key is lost, it will be difficult for a user to recover the original data. Trusted third parties are difficult to deploy in real-world applications and can easily become a bottleneck. Aiming at the above problems, we propose a data deduplication scheme based on double-layered encryption and key sharing, which does not require participation of any online trusted third party. On the premise of protecting data privacy, in order to improve the efficiency of deduplication, we recognize the popularity of data and adopt different encryption methods. For unpopular data with high privacy, a double-layered encryption mechanism is adopted to ensure the privacy of data. The inner layer is convergent encryption, the outer layer is symmetric encryption. The encryption key used for the outer layer encryption is sent to multiple key management servers by means of threshold secret sharing. In this way, the burden of key management on the user side is alleviated, and key sharing across users can be achieved. For popular data with low privacy, simple and efficient convergent encryption is adopted. The deduplication of unpopular data is achieved, an efficient method for implementing popularity conversion operation is designed. Therefore, for any data in our scheme, only one copy is stored in the cloud whether the data is popular or not, which effectively minimized the storage overhead for the cloud server. Security analysis is provided, in which the anti-attack capability of the scheme is discussed. Our scheme is not only resistant to internal attacks and external attacks, but also resistant to collusion attacks under certain conditions. Characteristic analysis and comparison show the advantages of scheme in terms of performance. Simulation experiments show that the proposed scheme is applicable and efficient. Compared with other existing deduplication schemes, our deduplication scheme has a significant performance advantage in terms of time overhead. Our scheme allows the deduplication of unpopular data, which further improves the efficiency of deduplication operations. Experimental results show that our scheme has better efficiency of deduplication and can greatly reduce the storage overhead of the cloud server.

Keywords encrypted data deduplication; double-layered encryption; key sharing; data popularity; cloud storage security

1 引 言

随着信息技术的不断发展与云存储服务的普及，越来越多的用户选择将数据外包到云服务器存储，这使得云服务器端存储的数据量呈爆炸式增长. 数据去重技术是一种高效的数据缩减技术，可提高云服务器的存储效率，降低网络带宽消耗. 数据去重也被称作重复数据删除(data deduplication). 对于明文数据，可采用随机抽样或提取散列值的方式进行数据重复性检测. 若用户新上传的数据与原存储数据相同，则删除重复数据，并告知该用户数据的访问链接,用户无需重复上传[1]. 数据去重技术通过删除冗余数据，对每个数据副本只存储一份，在云存储服务中得到广泛的应用.

但随着信息安全问题的不断出现，用户的安全意识提高，云数据的安全存储问题得到关注[2,3]. 为保护个人数据隐私，用户普遍将数据进行加密后，再上传到云服务器存储. 同一份数据经过不同用户加密，得到的密文不同，使得云服务器难以进行数据去重. 为实现加密数据的去重，Douceur 等人提出了收敛加密(CE, convergent encryption)[4]. CE 中使用的数据加密密钥由原数据经过散列计算得到，同一数据生成相同的密钥，经过确定性算法加密得到的密文相同. 收敛加密方案首次实现了对密文的去重，在云存储数据去重领域得到了广泛的应用[5-8]. 但是，CE 的密钥由原始数据计算得到，若敌手已知密文，即可通过穷举明文进行加密后与之对比，对明文数据进行猜测. 收敛加密不能满足语义安全要求，难以抵抗离线穷举攻击[9]. 在收敛加密的基础上，为明确安全目标，提供形式化安全模型，Bellare 等人提出消息锁加密(MLE, message-locked encryption)[5]. Bellare 等人在 MLE 的基础上提出交互式消息锁加密(iMLE, interactive message-locked encryption)，加密密钥由原始数据和系统参数共同计算得到[6].

为实现大规模加密文件去重，Wollongong 大学的穆怡教授团队提出了块级消息锁加密(BL-MLE, block-level message-locked encryption)[7]，在 MLE 的框架基础上，将文件级与块级数据去重相结合，降低去重粒度，进一步提高云服务器存储效率. 但这些基于 MLE 的改进方案，在本质上和 CE 相同，不能达到语义安全.

为抵抗现有针对 CE 的攻击，Puzio 等人提出了 ClouDedup 方案[10]，添加访问控制机制，并将数据加解密过程外包给可信第三方执行，保护数据隐私性. Bellare 等人提出 DupLESS 方案[11]，引入密钥服务器 KS(key server)生成数据加密密钥，可有效地抵抗蛮力攻击，但同时要求 KS 完全可信. 针对不同隐私程度的数据具有不同的安全需求，Stanek 等人首次提出了“数据流行度”的概念[12]. 根据隐私程度将用户数据划分为非流行数据和流行数据，并采用不同的加密算法保护. Puzio 等人提出 PerfectDedup 方案[13]，用户使用完美散列函数计算数据块标识，并借助可信第三方实现流行数据的去重. 但可信第三方在现实中部署困难，需依赖可信第三方的云数据去重方案安全性受限，实用性不高.

对于 CE 中收敛密钥的管理问题，广州大学的李进教授等人提出了 Dekey 方案[14]，借助秘密共享机制将收敛密钥保存到多个密钥管理服务器，实现加密数据去重中高效可靠的密钥管理. Stanek 等人在原有方案基础上提出了加强版数据安全去重方案[15]，引入门限收敛密码系统，保护非流行数据加密密钥的安全性. 该方案需要借助可信第三方提供索引服务，实现对流行数据的去重. 对于非流行数据，云服务器端仍保存有多份数据副本，存在数据冗余.

Liu 等人提出一种无可信第三方的加密数据安全去重方案[16]，用户上传数据时需与之前上传该数据的用户在线运行 PAKE(Password authenticated key exchange)协议进行密钥交换. 该方案要求参与协商的用户实时在线，不仅增加了通信开销，也降低了方案的实用性. Singh 等人提出一种安全的数据去重方案 DSSS，采用 POB 数字系统对用户数据进行隐私保护，并借助基于中国剩余定理的秘密共享机制，将加密密钥分布到多个密钥管理服务器[17]. 该方案使用多个服务器存储用户数据，很大程度上增加了服务器端的存储开销，也增加了用户与服务器间的通信开销. 西安电子科技大学的陈晓峰教授团队提出 dedupDUM 方案，支持动态的用户管理，使用预先验证的访问控制技术，验证用户身份的有效性[18]. 该方案采用重加密技术保护数据隐私，但云服务器端频繁的加解密操作一定程度上消耗了计算资源. 中国石油大学的范永开教授等人提出了一种隐私保护的数据去重方案[19]，依赖可信执行环境提供安全的密钥管理，保护敏感数据的隐私性. 西安电子科技大学的王保仓教授团队提出一种基于密钥共享的数据安全去重方案[20]，对于加密密钥的管理问题提出一种基于所有权证明的密钥共享方法，但需借助可信第三方实现. Hou 等人提出一种支持加密数据去重的云存储审计方案[21]，实现流行度转换阶段对用户数据的完整性审计.

针对以上问题，本文提出了一种基于双层加密和密钥共享的云数据安全去重方案，摆脱了可信第三方的束缚，实现云存储中加密数据的安全去重. 主要贡献如下：

（1）克服了对单一可信第三方的依赖，去重机制能够有效地抵抗内部攻击和外部攻击，而且具备一定的容错能力.

（2）在数据流行度转换阶段，进一步验证数据完整性，并能够以高效的方式实现流行度转换操作.

（3）实现了对非流行数据的去重，进一步提高了去重效率.

2　预备知识

2.1　Shamir秘密共享

秘密共享，是将一个秘密 S 拆分成 n 份子秘密 $\{S_1,S_2,\cdots,S_n\}$. 只有得到任意 k 份或更多子秘密才能恢复出原始秘密 S，而任意 $k-1$ 或更少的子秘密则无法得到关于 S 的任何信息，其中 $k\leqslant n$. Shamir 用代数学的方法给出了最早的门限秘密共享算法，利用有限域上的多项式将秘密进行拆分，并通过拉格朗日插值公式恢复原始秘密[22].

本文借助 SSSS(Shamir’s Secret Sharing Scheme)来实现不同用户间的密钥共享. SSSS-(n, k)中包含 2 个参数，n 为秘密共享的数量，k 为秘密恢复的门限值，且 $n>k>0$. 主要包含以下两个函数：

（1）$\{SK_1,SK_2,\cdots,SK_n\}\leftarrow Share(SK)$：密钥共享函数. 输入加密密钥 SK，输出 n 份同等长度的子密钥 $\{SK_1,SK_2,\cdots,SK_n\}$.

（2）$SK\leftarrow Recover\left(\left\{SK_{i_1},SK_{i_2},\cdots,SK_{i_k}\right\}\right)$：密钥恢复函数. 以任意 k 份子密钥作为输入，输出原始密钥 SK.

Shamir 秘密共享算法具有加法同态性质[23]，用 $\oplus$ 代表加法运算，则对于任意两个秘密 x 和 y，满足 $\{x_1\oplus y_1,x_2\oplus y_2,\cdots,x_n\oplus y_n\}\leftarrow Share(x\oplus y)$，其中

x_i、$y_i(1\leqslant i\leqslant n)$分别为 x、y 的子秘密.

2.2 对称加密

对称加密使用同一密钥对数据加密解密，主要包含以下三个函数：

（1）$SK\leftarrow KGen_{SE}(1^{\lambda})$：密钥生成函数. 以参数 1^{λ} 作为输入，生成加密密钥 SK.

（2）$C\leftarrow Enc_{SE}(SK,M)$：加密函数. 输入加密密钥 SK 和数据 M，输出加密后的密文 C.

（3）$M\leftarrow Dec_{SE}(SK,C)$：解密函数. 输入加密密钥 SK 和密文 C，输出原始数据 M.

2.3 收敛加密

收敛加密直接使用明文数据 M 的收敛结果 $H(M)$ 作为密钥，对明文数据进行加密，并使用密文的哈希值作为数据标识[6]. 收敛加密计算简单，执行效率高. 但密钥与明文数据具有直接的关系，不符合语义安全要求，易遭受离线穷举攻击. 即针对 M 的收敛密文 C，攻击者穷举 $\{M_i\}(i=1,2,\cdots)$，用 M_i 的收敛结果 $H(M_i)$ 加密数据得到密文 C_i. 通过对比 $C=C_i$ 是否成立，即可对明文数据 M 进行猜测.

收敛加密算法主要包含以下四个函数：

1）$K_C\leftarrow KGen_{CE}(M)$：密钥生成函数. 以数据 M 作为输入，生成其收敛密钥 K_C.

2）$C\leftarrow Enc_{CE}(K_C,M)$：加密函数. 用收敛密钥 K_C 加密数据 M，输出收敛密文 C.

3）$T_M\leftarrow TagGen(C)$：标签生成函数. 以收敛密文 C 作为输入，生成数据标签 T_M.

4）$M\leftarrow Dec_{CE}(K_C,C)$：解密函数. 用收敛密钥 K_C 解密 C，得到原始数据 M.

2.4 所有权证明

所有权证明（Proof of Ownership，PoW）协议用于解决使用一个短哈希值作为整个文件代理所产生的安全问题，是提高客户端数据去重方案安全性的有效措施[24]. 通过 PoW 协议，用户可以在不需发送整个文件的情况下，向云服务器证明其确实拥有对某个数据副本的所有权，很大程度上节省了网络带宽. 具体来说，PoW 是一种由证明者（用户）和验证者（云服务器）运行的交互式算法. 云服务器上已存储数据 M 的副本，并根据 M 的信息进行预计算，生成质询集合 $Challenge$ 和对应的回答集合 $Response$，采用挑战问答的形式对用户的数据所有权进行验证. 挑战阶段，云服务器随机选取未被使用的质询 $chall$ 发送给用户，其中 $chall\in Challenge$. 如果用户能计算并返回正确的回答，通过云服务器端的验证，则说明用户拥有 M 的所有权；否则，验证失败.

3 系统模型和定义

3.1 系统模型

本文方案的系统模型如图 1 所示，包含云服务器、用户和密钥管理服务器三类实体.

云服务器 CSP(Cloud storage provider)：主要提供云存储服务，并对上传相同数据的用户进行计数. 该计数用于数据流行度查询.

用户(User)：将数据上传至云服务器以节省本地存储空间. 为保护数据隐私，将加密后的数据上传至云服务器.

密钥管理服务器 KMS(Key management server)：借助秘密共享机制管理数据的加密密钥. 单个 KMS 上只存储加密密钥的一个子密钥，多个子密钥可用于恢复原加密密钥. KMS[j]表示编号为 j 的密钥管理服务器.

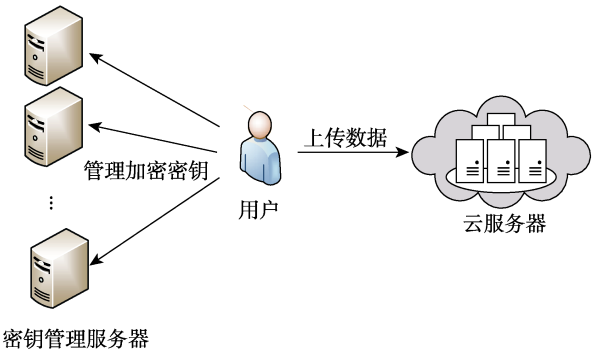

图 1 系统模型

3.2 威胁模型

在本文方案中，我们主要考虑以下两类攻击者：

（1）内部攻击者：指系统内部的敌手，主要指云服务器和密钥管理服务器. 我们认为云服务器是诚实且好奇的，可对其存储的用户数据进行任意访问. 我们认为密钥管理服务器是不完全可信的，它们之间存在合谋的可能，尝试获取用户密钥信息. 在本文方案中，我们认为云服务器与部分密钥管理服务器也可能合谋.

（2）外部攻击者：指系统外部的敌手，主要指恶意用户. 外部攻击者通过窃听公共信道获取部分上传数据或子密钥的相关信息，尝试获取云服务器上存储的用户数据的明文信息.

3.3 安全目标

本文方案主要有以下三个安全目标：

（1）非流行数据加密密钥的语义安全：借助第三方服务器管理非流行数据的外层加密密钥，无需用户间传递密钥，保证密钥的安全性，实现不同用户间密钥共享. 不需要用户保存加密密钥，而且具有一定的容错性. 去重方案应确保敌手无法获取加密密钥.

（2）数据的隐私性：去重方案应保证存放在云服务器上用户数据的隐私性. 敌手不应获得任何关于用户数据的明文信息.

（3）数据的完整性：去重方案应保证存储在云服务器上用户数据的完整性，尤其是在数据流行度发生转变时. 此外，去重方案应允许数据的所有者在下载数据时验证其完整性.

3.4　符号定义

本文中使用的符号及其定义如表 1 所示.

对于文件 F，当 $num_F < t$ 时，定义 F 为隐私程度较高的非流行数据；$num_F = t$ 时，F 由非流行数据转换为流行数据；当 $num_F > t$ 时，定义 F 为隐私程度不高的流行数据.

表 1　符号定义

符号	含义
t	数据流行度阈值
n	密钥管理服务器的数量
k	秘密恢复的门限值
num_F	成功上传 F 的合法用户数量
T_F	文件标签，作为 F 的唯一标识. $T_{F,j}$ 表示针对 KMS[j]的文件标签，作为 F 在 KMS[j]上的唯一标识
G_F	文件 F 对应密文. C_F^1 表示单层加密密文，C_F^2 表示双层加密密文
K_F	文件 F 的外层加密密钥. $K_{F,j}$ 表示存储在 KMS[j]上子密钥 $K_{F,j}$
$auxK$	辅助密钥，用于辅助生成文件外层加密密钥. $auxK_j$ 表示 KMS[j]上存储的辅助子密钥，初始值为 0
P	指针. P_F 代表数据指针，指向存储在 CSP 上 F 对应密文；$P_{K_{F,j}}$ 代表子密钥指针，指向存储在 KMS[j]上子密钥 $K_{F,j}$
PoW	所有权证明协议. PoW_F 代表由 CSP 发起的针对 F 的所有权证明协议

4　方　案

本文方案借助密钥管理服务器管理非流行数据的外层加密密钥，实现云存储中加密数据的安全去重. 通过划分数据流行度，对于隐私程度较高的非流行数据，采用双层加密机制进行保护. 对于隐私程度不高的流行数据，云服务器存储其收敛加密结果.

4.1　文件上传

用户 User 选择文件 F 上传到云服务器 CSP 存储时，首先对 F 进行收敛加密，得到收敛密文 C_F^1. 计算文件标签 T_F，并将其发送到 CSP 进行数据重复性检测. 若 T_F 不存在，则 F 为新文件，CSP 存储 T_F 并初始化 num_F=0，否则，F 为重复文件. CSP 根据上传 F 的合法用户数量 num_F 与数据流行度阈值 t 之间的关系，将文件上传过程分为以下三种情况：

（1）$num_F < t$，F 为隐私程度较高的非流行数据，此时为非流行文件上传，执行 Algorithm 1.1.

（2）$num_F = t$，定义为流行度转换，执行 Algorithm 1.2，即本次上传结束之后，F 由非流行数据转变为流行数据.

（3）$num_F > t$，F 为隐私程度不高的流行数据，此时为流行文件上传，执行 Algorithm 1.3.

文件上传的具体流程见 Algorithm 1.

Algorithm 1

1. User:
2. 　Compute convergent key $K_C \leftarrow KGen_{CE}(F)$
3. 　Compute convergent ciphertext $C_F^1 \leftarrow Enc_{CE}(K_C, F)$
4. 　Generate file tag $T_F \leftarrow TagGen(C_F^1)$
5. User → CSP: Send T_F to CSP
6. CSP：
7. 　IF T_F does not exist, THEN
8. 　　Store T_F and initialize num_F=0
9. 　IF $num_F < t$, THEN execute Algorithm 1.1
10. 　ELSE IF $num_F = t$ THEN execute Algorithm 1.2
11. 　ELSE execute Algorithm 1.3

4.1.1　非流行文件上传

$num_F < t$ 时，定义本次上传为非流行文件上传. 非流行文件上传的具体流程见 Algorithm 1.1.

用户与 n 个密钥管理服务器交互获取 F 的外层加密密钥 K_F 对应的子密钥指针. 首先，用户生成密钥 $SK \leftarrow KGen_{SE}(1^\lambda)$，并通过门限秘密共享机制将 SK 分发到 n 个密钥管理服务器. 对于每一个 KMS[j]，用户生成针对该服务器的文件标签 $T_{F,j} \leftarrow TagGen(C_F^1, j)$，计算 SK 的子密钥 $SK_j \leftarrow Share(SK)$，并将其发送到

KMS[j]. KMS[j]检测$T_{F,j}$是否存在，若$T_{F,j}$不存在，计算 F 对应子密钥 $K_{F,j}=SK_j\oplus auxK_j$ 并存储 $T_{F,j}$、$K_{F,j}$，更新 $auxK_j=SK_j$，其中 $auxK_j$ 为辅助子密钥. 否则，$T_{F,j}$ 和 $K_{F,j}$ 已存储在 KMS[j]. KMS[j]返回子密钥指针 $P_{K_{F,j}}$ 给用户. 该指针指向存储在 KMS[j]上的子密钥 $K_{F,j}$.

用户对 F 进行再加密，并将双层加密密文上传到 CSP 存储. 具体来说，用户根据子密钥指针到 KMS 上获取对应子密钥，运行密钥恢复函数得到外层加密密钥 $K_F\leftarrow Recover(K_{F,i_1},K_{F,i_2},\cdots,K_{F,i_k})$，并使用该密钥对 F 的收敛密文进行加密，得到双层加密密文 $C_F^2\leftarrow Enc_{SE}(K_F,C_F^1)$. 用户上传 C_F^2 到 CSP. CSP 查询 num_F，若 $num_F=0$，直接存储 C_F^2 并更新 $num_F=1$；否则，F 为后续上传，CSP 检测当前上传的 C_F^2 与已存储的 F 对应密文是否重复，若 C_F^2 为重复文件，则不存储 C_F^2，并更新 $num_F+=1$，否则存储 C_F^2 并记录该错误. CSP 返回数据指针 P_F 给用户，该数据指针 P_F 指向存储在 CSP 上 F 对应密文.

Algorithm 1.1

Case 1: $num_F<t$

1. User: Generate encryption key $SK\leftarrow KGen_{SE}(1^\lambda)$
2. FOR j = 1 to n DO:
3. User:
4. Generate file tag $T_{F,j}\leftarrow TagGen(C_F^1,j)$
5. Compute key share $SK_j\leftarrow Share(SK)$
6. User → KMS[j]: Send $T_{F,j}$, SK_j to KMS[j]
7. KMS[j]:
8. IF $T_{F,j}$ does not exist, THEN
9. $K_{F,j}=SK_j\oplus auxK_j$, and store $T_{F,j}$, $K_{F,j}$
10. update $auxK_j=SK_j$
11. ELSE $T_{F,j}$ and $K_{F,j}$ already exist
12. KMS[j] →User: Send key share pointer $P_{K_{F,j}}$ to user
13. END FOR
14. User:
15. Recover key $K_F\leftarrow Recover(K_{F,i_1},K_{F,i_2},\cdots,K_{F,i_k})$
16. Re-encrypt F $C_F^2\leftarrow Enc_{SE}(K_F,C_F^1)$
17. User → CSP: Send C_F^2 to CSP for storage
18. CSP:
19. IF $num_F=0$ THEN
20. Store C_F^2 and update $num_F=1$
21. ELSE IF C_F^2 is a duplicate file, THEN
22. Discard C_F^2 and $num_F+=1$
23. ELSE, store C_F^2 and record the error
24. Return data pointer P_F to user
25. User: Store $\{T_F,P_F,P_{K_F},K_C\}$

用户存储 $\{T_F,P_F,P_{K_F},K_C\}$，其中 $P_{K_F}=\{P_{K_{F,1}},P_{K_{F,2}},\cdots,P_{K_{F,n}}\}$.

4.1.2 流行度转换

$num_F=t$ 时，定义为流行度转换. 流行度转换阶段的具体流程见 Algorithm 1.2.

本次上传结束之后，F 由非流行数据转变为流行数据. 数据流行度发生转变，F 转变为隐私程度较低的流行数据，采用简单高效的收敛加密. 用户直接上传收敛密文 C_F^1 到 CSP，不需再借助第三方服务器管理加密密钥. CSP 收到后先验证标签的一致性. 若 $T_F=TagGen(C_F^1)$，则用 C_F^1 替换已存储的 F 对应密文，并返回数据指针 P_F 给用户. 用户存储 $\{T_F,P_F,K_C\}$.

Algorithm 1.2

Case 2: $num_F=t$

1. User → CSP: Send C_F^1 to CSP for storage
2. CSP:
3. IF $T_F=TagGen(C_F^1)$ THEN
4. Replace the original ciphertext with C_F^1
5. $num_F+=1$ and return data pointer P_F to user
6. ELSE return ERROR
7. User: Store $\{T_F,P_F,K_C\}$

4.1.3 流行文件上传

$num_F>t$ 时，定义为流行文件上传. 流行文件上传的具体流程见 Algorithm 1.3.

F 为流行文件，用户不再上传数据，而是与 CSP 交互获取数据指针. 用户通过与 CSP 运行文件所有权证明协议 PoW_F 证明其对 F 的所有权. PoW_F 验证通过之后，CSP 更新 $num_F+=1$，并返回数据指针 P_F 给用户. 用户存储 $\{T_F,P_F,K_C\}$.

Algorithm 1.3

Case 3: $num_F>t$

1. CSP →User: Send PoW verification demand to user
2. User running PoW_F with CSP
3. IF verification successful THEN
4. CSP: $num_F+=1$ and return data pointer P_F to user
5. User: Store $\{T_F,P_F,K_C\}$
6. ELSE return ERROR

4.2 文件下载

用户 User 下载文件 F，首先根据数据指针 P_F 到

云服务器 CSP 上下载 F 对应密文 C_F，然后解密得到原始数据. 文件下载的具体流程见 Algorithm 2. 用户解密数据分以下两种情况：

（1）若用户保存有 $\{T_F, P_F, P_{K_F}, K_C\}$，说明上传 F 时，F 为非流行数据. 用户根据密钥指针 $P_{K_F} = \{P_{K_{F,1}}, P_{K_{F,2}}, \cdots, P_{K_{F,n}}\}$，到 n 个密钥管理服务器上获取子密钥 $\{K_{F,1}, K_{F,2}, \cdots, K_{F,n}\}$. 以任意 k 个子密钥作为输入运行密钥恢复函数得到加密密钥，$K_F \leftarrow Recover(K_{F,i_1}, K_{F,i_2}, \cdots, K_{F,i_k})$. 用户使用 K_F 解密数据得到 $M_F \leftarrow Dec_{SE}(K_F, C_F)$. 若 $T_F = TagGen(M_F)$，则 M_F 为 F 的收敛密文. 进一步使用收敛密钥 K_C 解密数据得到原始文件 $F = Dec_{CE}(K_C, M_F)$. 若 $T_F = TagGen(C_F)$，说明成功上传 F 的合法用户数量已超过流行度阈值，此时 F 为流行数据，CSP 上存储的 F 对应密文已被替换为收敛密文. 该情况下，下载的密文 C_F 即为 F 的收敛密文，直接使用收敛密钥 K_C 解密数据得到原始文件 $F = Dec_{CE}(K_C, C_F)$. 否则，返回错误.

（2）若用户保存有 $\{T_F, P_F, K_C\}$. 首先验证标签的一致性，若 $T_F = TagGen(C_F)$，则直接使用收敛密钥 K_C 解密数据得到原始文件 $F = Dec_{CE}(K_C, C_F)$.

Algorithm 2

1. User → CSP: Downloading ciphertext C_F from CSP according to P_F
2. User:
3. IF have $\{T_F, P_F, P_{K_F}, K_C\}$ THEN
4. FOR j =1 to n DO:
5. User→KMS[j]: Obtain key share $K_{F,j}$ from KMS [j] according to $P_{K_{F,j}}$
6. END FOR
7. Recover key $K_F \leftarrow Recover(K_{F,i_1}, K_{F,i_2}, \cdots, K_{F,i_k})$
8. Compute $M_F \leftarrow Dec_{SE}(K_F, C_F)$
9. IF $T_F = TagGen(M_F)$ THEN
10. Compute $F = Dec_{CE}(K_C, M_F)$
11. ELSE IF $T_F = TagGen(C_F)$ THEN
12. Compute $F = Dec_{CE}(K_C, C_F)$
13. ELSE return ERROR
14. END
15. ELSE IF have $\{T_F, P_F, K_C\}$ THEN
16. IF $T_F = TagGen(C_F)$ THEN
17. Compute $F = Dec_{CE}(K_C, C_F)$
18. ELSE return ERROR
19. END
20. ELSE return ERROR

5　安全性分析

根据 3.2 节提出的威胁模型，本文方案主要考虑来自内部敌手和外部敌手的攻击. 内部敌手主要是指云服务器和密钥管理服务器. 外部敌手主要指恶意用户，其主要目的是非法获取 CSP 上存储的用户数据. 设置系统的安全参数为 s，且 $n > k > s$. 假设系统中最多同时存在 s 个密钥管理服务器参与合谋. 本节从以下三方面分析方案的安全性.

5.1　非流行数据加密密钥的语义安全

本文方案可有效地保证非流行数据加密密钥的语义安全，而且具有一定的容错能力. 在本文提出的去重方案中，非流行数据的外层加密密钥借助秘密共享机制存储在 n 个密钥管理服务器上. 根据门限秘密共享机制的安全性[25]，存放在密钥管理服务器上的加密密钥是语义安全的. 用户不需保存加密密钥，只需保存存储在 n 个 KMS 上子密钥对应指针. 而且密钥恢复阶段具有一定的容错能力，有效地避免了因加密密钥丢失而导致数据丢失的情况. 即使用户丢失部分子密钥指针，只要用户拥有 k 数量的子密钥指针，仍可恢复加密密钥.

5.2　数据的隐私性

方案保证存储在 CSP 上用户数据的隐私性，敌手不应获取任何数据明文. 由于用户数据是加密后上传，CSP 在不知密钥的情况下无法获得关于数据的任何明文信息. 单个 KMS 只存放了数据加密密钥的一个子密钥，门限秘密共享机制的安全性保证了单个 KMS 无法获知关于加密密钥的任何信息. 即使 CSP 与多个 KMS 合谋，在参与者数量小于系统安全参数 s 的情况下，也无法恢复出加密密钥. 因此，本文方案能有效地抵抗内部攻击，保证存储在 CSP 上用户数据的隐私性. 接下来，主要针对外部攻击进行安全性分析，建立一个安全模型游戏 *Game*. 在该游戏中，除了云服务器、密钥管理服务器和用户，同时还存在一个多项式时间的攻击者 A. 假设在游戏的交互中，A 凭借其攻击能力最多可攻陷 s 个 KMS 获取其存储的子密钥信息. 我们考虑以下情况，A 通过窃听信道可能获取部分数据标签，并且 A 可以计算任意数据的标签.

引理 1.　PoW 协议的安全性[26]. 在没有原始数据 M 的情况下，仅通过与数据相关的部分信息 *Info(M)*，欺骗服务器成功通过 PoW 验证的概率是可忽略的. 我们用 ε 表示可忽略值，即

$$Pr[Pow_M \mid Info(M)] \leqslant \varepsilon \tag{1}$$

引理 2. 密钥恢复函数的安全性. 基于门限秘密共享机制的安全性[25]，通过运行密钥恢复函数，任意 k 个子密钥可恢复原始密钥；但少于 k 个子密钥不能恢复出原始密钥. 即

$$\{K_{F,i_1},K_{F,i_2},\cdots,K_{F,i_k}\}\xrightarrow{recover}K_F \tag{2}$$

引理 3. 哈希函数的抗碰撞性. 安全的哈希函数 $H:\{0,1\}^*\to\{0,1\}^l$，对于 $\forall D_1,D_2\in\{0,1\}^*$ 且 $D_1\neq D_2$，使得 $H(D_1)=H(D_2)$ 的概率是可忽略的. 即

$$Pr[H(D_1)=H(D_2)\,|\,D_1\neq D_2]\leqslant\varepsilon \tag{3}$$

定理 1. 本文方案能有效地抵抗外部攻击，保证存储在 CSP 上用户数据的隐私性. 即 A 在已知数据标签 T_M 但没有原始数据 M 的情况下，赢得游戏 *Game* 获取数据明文的概率是可忽略的.

$$Pr[A_{wins}]\leqslant\varepsilon$$

证明. 根据 4.1 节文件上传部分，A 想要获取数据，需要与 CSP 交互获取数据密文指针. 若 M 为非流行数据，根据 4.1.1 节非流行文件上传流程，A 在没有 M 的情况下不能正确上传其对应双层加密密文，即 A 无法从获取 CSP 上存储的数据密文对应指针. 若 M 为流行数据，根据引理 1，$Pr[Pow_M\,|T_M]\leqslant\varepsilon$，即 A 在仅拥有 T_M 的情况下，通过 *PoW* 验证的概率是可忽略的. 因此，A 仅凭借数据标签获取 CSP 上数据密文赢得 *Game* 的概率是可忽略的，即 $Pr[A_{wins}\,|T_M]\leqslant\varepsilon$.

即使 A 已获取密文 C，但 CSP 上存储的数据密文有两种形式：

a) 非流行数据的双层加密密文；

b) 流行数据的收敛密文.

A 在没有密钥且不知密文形式的情况下，想要解密 C 分以下两种情况：

（1）考虑 C 为双层加密密文. 根据假设条件，A 至多可获取 s 个子密钥 $\{K_{F,i_1},K_{F,i_2},\cdots,K_{F,i_s}\}$. 根据引理 2，$\{K_{F,i_1},K_{F,i_2},\cdots,K_{F,i_s}\}\xrightarrow{recover}K_F$，A 根据已获取的子密钥数量不足够恢复 K_F，即不能对 C 进行解密.

（2）考虑 C 为单层收敛密文. 根据引理 3，A 在没有 M 的情况下，难以计算出收敛密钥 K_C，即不能解密得到数据明文.

CSP 上存储数据密文形式的不可区分性，A 通过解密 C 获取数据明文，赢得游戏 *Game* 的概率是可忽略的，即 $Pr[A_{wins}\,|C]\leqslant\varepsilon$. 因此，本文方案能有效地抵抗外部攻击，保证存储在 CSP 上用户数据的隐私性.

进一步，本文方案可有效地抵抗边信道攻击[27]，保护用户数据隐私. 对于非流行数据，用户与 n 个 KMS 交互获取子密钥. 为避免辅助密钥 *auxK* 的泄露，我们采用“一次一更新”的方式对其进行更新，即每次用过之后即对 *auxK* 进行更新，同时也使得用户对返回的子密钥在计算上不可区分. 用户通过密钥恢复函数得到外层加密密钥，对数据进行加密后将双层加密密文上传到 CSP. 基于服务器端的去重，使得用户难以辨别上传的文件是否重复，从而有效地抵抗边信道攻击.

5.3 数据的完整性

对于非流行数据，本文方案可抵抗恶意用户的伪造攻击，有效地避免恶意用户通过上传虚假数据来破坏数据完整性. 方案中基于密文去重，当恶意用户上传与标签不匹配的密文时，CSP 会立即发现并记录异常. 即使恶意用户作为 F 的初始上传者生成标签 T_F，而上传另一个文件 F' 的双重加密密文，导致标签 T_F 与密文 $C_{F'}^2$ 不匹配. 在该文件的后继的上传过程中，CSP 也会发现并记录该异常，而不影响该存储文件对于后继用户的可用性. 该情况下，尽管 CSP 无法立即分辨当前用户是否为恶意用户，但若有更多的后继用户上传双层加密密文，CSP 可以根据上传某一个文件副本的数量来判断用户是否上传虚假数据，并进行记录，从而可以有效地检测恶意用户的伪造攻击行为.

在流行度转换阶段，CSP 可以通过检测标签的一致性，验证用户上传数据的完整性. 在 4.1.2 节流行度转换阶段，用户需上传 F 对应的收敛密文 C_F^1 替换 CSP 上原有密文. CSP 通过计算 $T_F'\leftarrow TagGen(C_F^1)$，与存储的 T_F 对比，验证用户上传数据的完整性.

定理 2. CSP 上存储数据的可验证性. 数据的合法用户在下载数据阶段，可以通过检测标签的一致性，验证其存储数据的完整性.

证明. 由 4.2 节中文件下载部分. 合法用户通过数据指针下载存储在 CSP 上 F 对应密文 C_F. 该用户对密文进行解密过程中即可验证数据完整性.

（1）若 F 为非流行数据，用户首先根据子密钥指针到 n 个 KMS 上获取子密钥 $\{K_{F,1},K_{F,2},\cdots,K_{F,n}\}$，运行密钥恢复函数获取外层加密密钥 K_F，用该密钥解密得到 $M_F\leftarrow Dec_{SE}(K_F,C_F)$. 本文方案中对非流行数据采用双层加密，所以解密得到的 M_F 应为 F 的收敛密文. 通过计算标签 $T_F'\leftarrow TagGen(M_F)$，并对比 T_F' 与本地存储的数据标签 T_F 是否相等，来验

证下载数据的完整性.

（2）若 F 为流行数据，则 C_F 即为 F 的收敛密文. 直接计算标签 $T_F' \leftarrow TagGen(C_F)$，并与本地存储的标签 T_F 进行对比. 若 $T_F' = T_F$，则表明数据完整性未遭破坏.

6　仿真实验与性能分析

我们分别模拟实现了云服务器端、密钥管理服务器端和用户端. 我们使用两台戴尔 OptiPlex 5060 台式计算机，配备有 2.10GHz Intel(R) Core(TM) i5-8500T 四核处理器和 8GB 运行内存，运行操作系统为 Windows 10，分别模拟云服务器端和密钥管理服务器端. 我们使用一台 Lenovo s40-70 计算机模拟用户端.

本节主要对提出方案进行性能分析，并通过仿真实验，测试方案性能和去重效率. 将本文方案与现有去重方案对比，分析说明本文方案在性能和去重效率方面的优势. 本节的主要内容包括方案的性能分析、计算时间开销、去重效率、门限秘密共享机制编码和解码时间四部分.

6.1　性能分析

将本文方案与 Clou Dedup 方案[10]、Perfect Dedup 方案[13]、Dekey 方案[14]、Stanek 方案[15]在方案特点方面进行对比，结果如表 2 所示. 本文方案无需可信第三方，用双层加密保护数据隐私，划分数据流行度并实现了对非流行数据的去重，而且具有容错性.

表 2　方案特点对比

	Clou Dedup	Dekey	Perfect Dedup	Stanek	Our scheme
无需可信第三方	×	✓	×	×	✓
双层加密	×	×	×	✓	✓
划分数据流行度	×	×	✓	✓	✓
非流行数据去重	×	×	×	×	✓
容错性	×	✓	×	×	✓

我们从理论上对方案[13]、方案[15]和本文方案在性能方面进行分析和对比，主要包括通信开销和用户端存储开销，对比结果如表 3 所示. 对于非流行数据 M，C_T、C_C 和 C_K 分别表示数据标签体积、密文体积和密钥体积. 数据指针和子密钥指针的体积较小，在此忽略不计. 本文方案在上传数据阶段，针对 CSP 和 n 个 KMS 上传数据标签，并上传子密钥到 n 个 KMS，上传数据时的通信开销为 $(n+1)C_T + C_C + nC_K$. 下载数据时通信开销为 $C_C + nC_K$. 根据对比结果可知，本文方案在上传数据和下载数据两个阶段，因引入密钥管理服务器产生了额外的通信开销. 但密钥体积 C_K 远小于密文体积 C_C，传递密钥在通信开销中所占比例较小. 在实际应用中，可通过控制密钥管理服务器的数量 n，降低对系统性能产生的影响. 在存储开销方面，用户端只需保存数据标签和收敛密钥，相比方案[13]和方案[15]，本文方案具有明显的优势.

表 3　方案性能对比

方案	通信开销		存储开销
	上传数据	下载数据	
方案[13]	$2C_T+C_C+C_K$	$C_T+C_C+2C_K$	$2C_T+C_K$
方案[15]	$2C_T+C_C+C_K$	$C_T+C_C+C_K$	$2C_T+2C_K$
本文方案	$(n+1)C_T+C_C+nC_k$	C_C+nC_k	C_T+C_K

6.2　计算时间开销

为测试方案性能，我们使用 SQL Server2017 数据库存储文件与用户的信息，采用 C#语言编写应用程序模拟去重过程. 方案的实现阶段，使用开源代码[28]实现密钥共享，借助 OPENSLL 函数库[29]实现方案中所需的密码算法. 实现细节方面，使用 SHA-1 哈希函数生成数据标签，SHA-256 用于生成收敛密钥，并采用 256bit 强度的 AES 算法实现非流行数据的外层加密. 传输数据过程中，限制网络传输带宽为 10Mbps.

我们对方案的计算时间开销进行测试，并与方案[13]和方案[15]进行对比. 测试的主要过程包含标签生成、密钥生成、密钥加密、数据加密和数据上传. 根据方案特点，我们将 100MB 体积的文件上传至 CSP，分以下三种情况进行模拟实验：

（1）非流行文件上传，即上传文件为非流行文件. 该情况下的测试结果如图 2 所示. 方案[13]对数据进行对称加密. 方案[15]和本文方案对数据进行双层加密. 对于外层加密密钥的加密，本文方案借助 KMS 存储其子密钥. 方案[15]通过门限收敛密码将其加密并上传到 CSP 存储，在计算时间上大于本文方案.

（2）流行度转换，即上传的文件在本次上传结束后由非流行数据转变为流行数据. 该情况下的测试结果如图 3 所示. 方案[13]需当前用户上传收敛密文，并由 CSP 删除其他数据副本. 方案[15]需由 CSP 解密出外层加密密钥，并执行外层密文的解密操作. 本文方案只需当前用户上传收敛密文替换 CSP 上原有密文，无需执行其他操作. 相对于方案[13]和方案[15]，本文方案能够以高效的方式实现流行度转换操作.

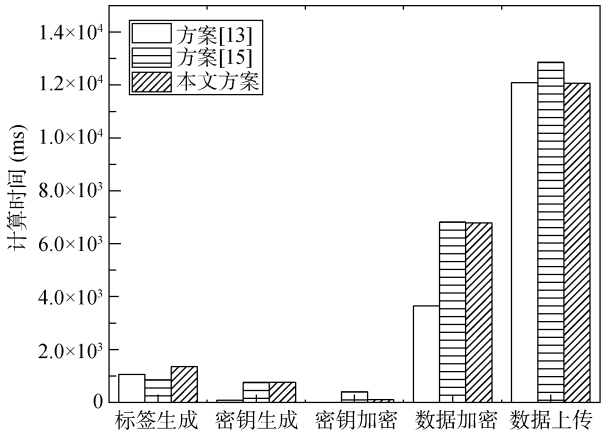

图 2 非流行文件上传情况

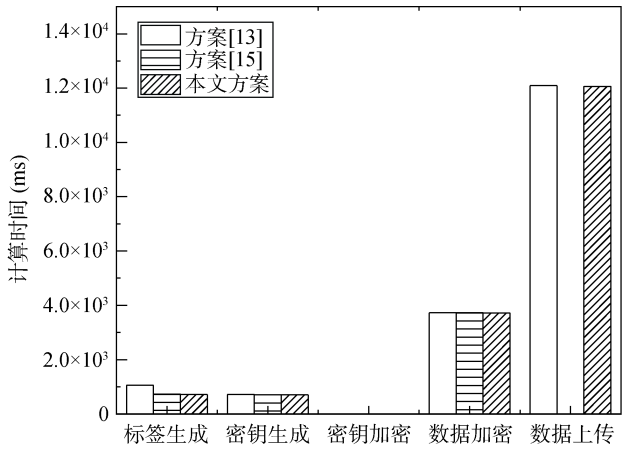

图 3 流行度转换情况

(3)流行文件上传,即上传文件为已存储在 CSP 上的流行文件. 该情况下三种方案均要求用户对数据进行收敛加密，但无需重复上传数据，测试结果如图 4 所示.

对三种情况的总时间开销统计结果如图 5 所示. 情况 1(非流行文件上传)下需要用户加密并上传数据，时间开销较大. 情况 2(流行度转换)中本文方案和方案[13]需要用户上传收敛密文，用户上传数据导致总时间开销大于方案[15]. 但是,相比方案[15]，本文提出方案能够以简单高效的方式实现流行度转换操作. 情况 3(流行文件上传)，三种方案均执行数据去重，总时间开销较小. 整体来看，本文方案不会产生额外的计算时间开销.

6.3 去重效率

本文中定义去重效率为用户需上传到 CSP 存储的数据中重复数据体积占总数据体积的比例. 为模拟真实的云存储数据去重场景,系统初始化 8000 个不同的文件，文件体积随机，上限为 500MB. 文件初始化过程借助文件生成器实现. 预先随机选取 2000 个不同的文件上传到 CSP，并初始化每个文件的合法用户数量为随机值. 设定流行度阈值为 5. 随机选择一定数量的文件上传到 CSP，根据不同的去重方案，统计数据去重情况，并计算去重效率.

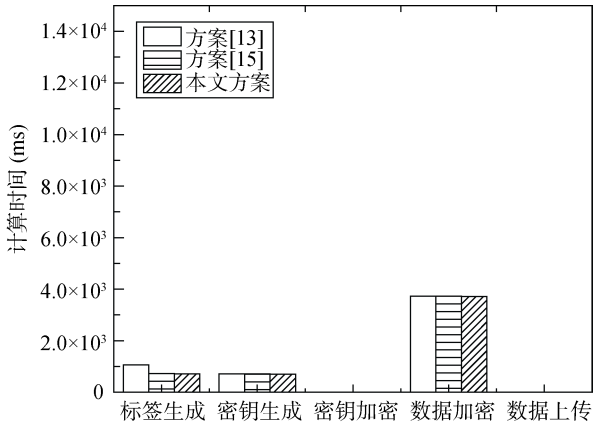

图 4 流行文件上传情况

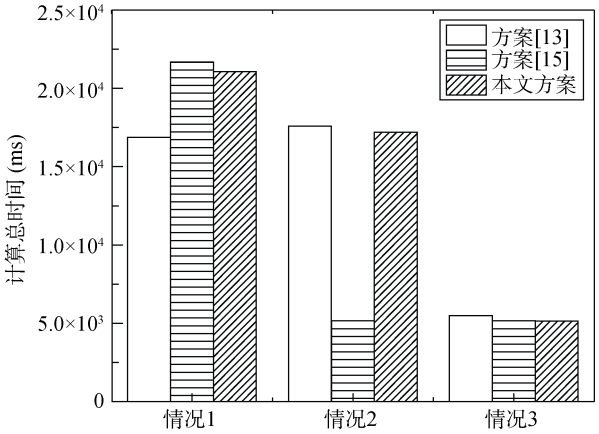

图 5 总时间开销对比

将本文方案与方案[13]和方案[15]进行对比，实验结果如图 6 所示. 三个方案都基于数据流行度划分，方案[13]和方案[15]都仅对流行数据执行去重操作，对于非流行数据，云服务仍保存多份数据副本. 方案[13]基于数据块去重，去重效率高于方案[15]. 本文方案基于文件级去重，不仅对流行数据执行去重操作，而且实现了对非流行数据的去重. 因此，在本文方案中，无论数据是否流行，云服务器端只存储一份数据副本，数据去重效率高于其他两个去重方案.

6.4 门限秘密共享机制的编码和解码时间

本文方案借助门限秘密共享机制，使用多个 KMS 管理非流行数据的外层加密密钥，以实现不同用户间的密钥共享. 实验通过改变子密钥数量 n 和门

限值 k 来测试门限秘密共享机制的编码和解码时间.

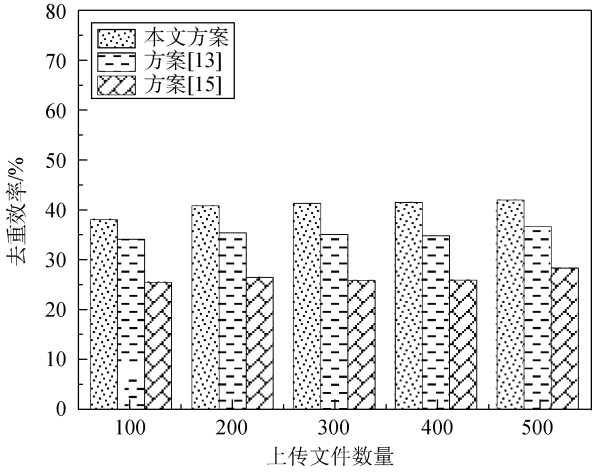

图 6　去重效率对比

编码和解码时间随 n,k 值变化情况如图 7 所示. 根据实验结果可知，编码和解码时间随着 n，k 值的增大而增加. 但是，门限秘密共享机制的编码和解码时间都是微秒级别的，与去重方案中数据加密和数据上传的时间相比，用于密钥共享的时间是可以忽略不计的. 因此，方案借助秘密共享机制来管理加密密钥，实现不同用户间的密钥共享，在计算时间上几乎不影响方案性能.

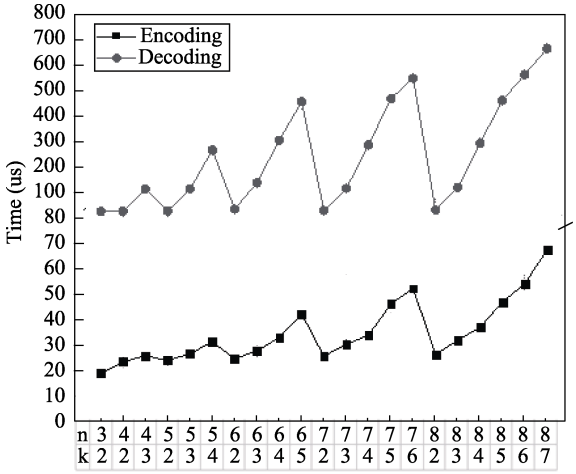

图 7　编码和解码时间

7　总　结

本文提出了一种基于双层加密和密钥共享的云数据去重方案，可实现云存储中加密数据的安全去重. 设定流行度阈值，将用户数据进行流行度划分. 对于非流行数据，采用双层加密保护数据隐私. 内层为收敛加密，外层为对称加密，并使用多个密钥管理服务器管理外层加密密钥，实现上传相同数据的不同用户间的密钥共享. 不仅减轻了用户端管理加密密钥的负担，而且具有一定的容错能力. 对于流行数据，直接采用收敛加密. 在数据流行度转换阶段，能够以高效的方式实现流行度转换操作，使得去重方案更具有实用性. 如何进一步在不借助任何第三方服务器的情况下，实现云存储中加密数据的安全去重，是我们接下来要研究的问题.

参　考　文　献

[1] Xiong Jin-Bo, Zhang Yuan-Yuan, et al. Research progress on secure data deduplication in cloud. Journal on Communications, 2016, 37(11): 169-180 (in Chinese)
(熊金波, 张媛媛, 李凤华, 等. 云环境中数据安全去重研究进展. 通信学报, 2016, 37(11): 169-180)

[2] Feng Chao-Sheng, Qin Zhi-Guang, Yuan Ding. Techniques if secure storage for cloud data. Chinese Journal of Computers, 2015, 38(1): 150-163 (in Chinese)
(冯朝胜, 秦志光, 袁丁. 云数据安全存储技术. 计算机学报, 2015, 38(1): 150-163)

[3] Tan Shuang, Jia Yan, Han Wei-Hong. Research and development of provable data integrity in cloud storage. Chinese Journal of Computers, 2015, 38(1): 164-177 (in Chinese)
(谭霜, 贾焰, 韩伟红. 云存储中的数据完整性证明研究及进展. 计算机学报, 2015, 38(1): 164-177)

[4] Douceur J R, Adya A, Bolosky W J, et al. Reclaiming space from duplicate files in a serverless distributed file system//Proceedings of the 22nd International Conference on Distributed Computing Systems. Vienna, Austria, 2002: 617-624

[5] Bellare M, Keelveedhi S, Ristenpart T. Message-locked encryption and secure deduplication//Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques. Athens, Greece, 2013: 296-312

[6] Bellare M, Keelveedhi S. Interactive message-locked encryption and secure deduplication//Proceedings of the IACR International Workshop on Public Key Cryptography. Berlin, Germany, 2015: 516-538

[7] Chen R M, Mu Y, Yang G M, et al. BL-MLE: Block-level message-locked encryption for secure large file deduplication. IEEE Transactions on Information Forensics and Security, 2015, 10(12): 2643-2652

[8] Zhang Shu-Guang, Xian He-Qun, Liu Hong-Yan, et al. Research on encrypted deduplication method based on offline key transfer in cloud storage environment. Netinfo Security, 2017(7): 66-72 (in Chinese)
（张曙光, 咸鹤群, 刘红燕, 等. 云存储环境中基于离线密钥传递的加密重复数据删除方法研究. 信息网络安全, 2017(7): 66-72）

[9] Drew Perttula and Attacks on Convergent Encryption, https://tahoe-lafs.org/hacktahoelafs/drew_perttula.html 2019,9,10

[10] Puzio P, Molva R, Önen M, et al. ClouDedup: Secure deduplication with encrypted data for cloud storage// Proceedings of the 2013 IEEE 5th International Conference on Cloud Computing Technology and Science. Bristol, UK, 2013, 1: 363-370
[11] Bellare M, Keelveedhi S, Ristenpart T. DupLESS: Server-aided encryption for deduplicated storage//Proceedings of the 22nd USENIX conference on Security. Washington, USA, 2013: 179-194
[12] Stanek J, Sorniotti A, Androulaki E, et al. A secure data deduplication scheme for cloud storage//Proceedings of the 18th International Conference on Financial Cryptography and Data Security. Christ Church, Barbados, 2014: 99-118
[13] Puzio P, Molva R, Önen M, et al. PerfectDedup: Secure data deduplication//Proceedings of the 10th International Workshop on Data Privacy Management. Vienna, Austria, 2015: 150-166
[14] Li J, Chen X F, Li M Q, et al. Secure deduplication with efficient and reliable convergent key management. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(6): 1615-1625
[15] Stanek J, Kencl L. Enhanced secure thresholded data deduplication scheme for cloud storage. IEEE Transactions on Dependable and Secure Computing, 2018, 15(4): 694-707
[16] Liu J, Asokan N, Pinkas B. Secure deduplication of encrypted data without additional independent servers//Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. Denver, USA, 2015: 874-885
[17] Priyanka S, Nishant A, Balasubramanian R. Secure data deduplication using secret sharing schemes over cloud. Future Generation Computer Systems, 2018, 88: 156-167
[18] Yuan H R, Chen X F, Jiang T, et al. DedupDUM: Secure and scalable data deduplication with dynamic user management. Information Sciences, 2018, 456: 159-173
[19] Fan Y K, Lin X D, Liang W, et al. A secure privacy preserving deduplication scheme for cloud computing. Future Generation Computer Systems, 2019, 101: 127-135
[20] Wang L, Wang B C, Song W, et al. A key-sharing based secure deduplication scheme in cloud storage. Information Sciences, 2019, 504:48-60
[21] Hou H Y, Yu J, Hao R. Cloud storage auditing with deduplication supporting different security levels according to data popularity. Journal of Network and Computer Applications, 2019, 134: 26-39
[22] Shamir A. How to share a secret. Communications of the ACM, 1979, 22(11): 612-613
[23] Benaloh J C. Secret sharing homomorphisms: keeping shares of a secret secret. Advances in Cryptology, 1987, 251-260
[24] Pietro R D, Sorniotti A. Proof of ownership for deduplication systems: A secure, scalable, and efficient solution. Computer Communications, 2007, 82:71-82
[25] Blakley G R, Meadows C. Security of ramp schemes//Advances in Cryptology, Proceedings of CRYPTO '84. Santa Barbara, USA, 1984, 242-268
[26] Halevi S, Harnik D, Pinkas B, et al. Proofs of ownership in remote storage systems//Proceedings of the 18th ACM Conference on Computer and Communications Security. Chicago, USA, 2011: 491-500
[27] Harnik D, Pinkas B, Shulman-Peleg A. Side Channels in cloud services: deduplication in cloud storage. IEEE Security & Privacy, 2010, 8(6):40-47
[28] CryDeS/ShamirSecretShareCS,https://github.com/CryDeS/ShamirSecretShareCS 2019-10-5
[29] Hu X, Qin Z, Zhang H, et al. Research and improved implementation of AES algorithm in OpenSSL. Microcomputer Information, 2009, 25(12): 83-85

GAO Wen-Jing, Ph.D. candidate. Her main research interests include cryptography and cloud security.

XIAN He-Qun, Ph.D., associate professor. His main research interests include cryptography, cloud security, and network security.

CHENG Run-Hui, M.S. His main research area is information security.

Background

In the Internet era, users generally upload data to the CSP (cloud storage provider) for storage or backup. In order to conserve network bandwidth and storage overhead of the CSP, data deduplication technology is widely used. With the rapid development of Internet technology, the data security problem of the CSP has become increasingly severe, and users pay more and more attention to data security. To protect data privacy, most users upload encrypted data to the CSP. Data encryption brings great challenges to data deduplication technology. The deduplication of encrypted data in cloud storage has become a difficult problem.

Most existing deduplication schemes for encrypted data need to rely on a trusted third party to achieve secure data deduplication. Most of the schemes based on popularity division only perform deduplication on popular data, and there is still a large amount of redundant data on the CSP. This paper focuses on the security and efficiency compatibility of encrypted data deduplication, and proposes a data deduplication scheme based on double-layered encry-

ption and key sharing. Our scheme does not require a trusted third party, and on the premise of ensuring data security, it realizes deduplication of unpopular data.

Cloud storage is a classical model of data storage in which the digital data is stored in logical pools. During its development, cloud storage has become a convenient and cost efficient way for companies and individual users to outsource data. However, although cloud storage has all sorts of advantages, there still exists some barriers or weaknesses that casually put a brake on the development of cloud storage. One of the most striking weakness is massive duplicated data, which are produced as different users may upload groups of identical data to the cloud, which significantly raises the storage costs of cloud servers. In this case, cloud service providers eagerly seek efficient ways to address the problem.

We published several papers in this research area, for example, “Secure encrypted data deduplication method based on offline key distribution” published in the Journal of Software, “Verifiable secure data deduplication based on user-defined security requirements” published in the Journal of Computer Research and Development, and so on. We also have some papers that have been accepted.

Our research is supported by the project of the National Natural Science Foundation (61702294), Shandong Provincial Natural Science Foundation (ZR2019MF058), and the Open Project Program of the State Key Laboratory of Information Security (2020-MS-09).