基于阈值重加密的抗边信道攻击云数据安全去重方法

唐鑫[1]，周琳娜[2]，单伟杰[1]，刘丹[1]

（1. 国际关系学院信息科技学院，北京 100091；2. 北京邮电大学网络空间安全学院，北京 100084）

摘 要：针对加密云数据阈值去重中的安全性和效率问题，提出一种基于阈值重加密的抗边信道攻击云数据安全去重方法。设计了一种轻量级的阈值重加密机制，将用户端的密文分割转变为密钥分割，并且把二次加密转移到云端执行，从而大大减少了用户端的计算开销。所提机制允许用户从一次加密密文和重加密密文中均可解密出明文，从而避免了对同一文件多次加密的开销。同时，所提方法支持云服务提供商和用户端双向的数据完整性验证，直接确保密文副本和用户端明文数据的对应性。实验结果表明，所提方法大大降低了用户端的计算开销，且同时取得了较好的云端存储性能。

关键词：阈值去重；重加密；边信道攻击；后验证

中图分类号：TP302

文献标识码：A

doi: 10.11959/j.issn.1000−436x.2020103

Threshold re-encryption based secure deduplication method for cloud data with resistance against side channel attack

TANG Xin[1], ZHOU Linna[2], SHAN Weijie[1], LIU Dan[1]

1. School of Information Science and Technology, University of International Relations, Beijing 100091, China
2. School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100084, China

Abstract: For security and efficiency problems in threshold based deduplication for cloud data, a novel method based on threshold re-encryption was proposed to deal with side channel attacks. A lightweight threshold re-encryption mechanism was presented to transfer the secondary encryption to the cloud for execution and allow clients to generate ciphertext based on key segmentation instead of ciphertext segmentation, both of which largely reduce computational overhead of clients. Also, the proposed mechanism enables clients to decrypt from both one-time encrypted and re-encrypted ciphertext, thus avoiding the overhead of redundant encryption of the same file. Mutual integrity verification between cloud service provider and clients was also supported by the proposed method, which directly ensured the correctness of the correspondence between ciphertext and plaintext on client side. Experiments show that the proposed method not only largely reduces the computational overhead on client side, but also achieves superior storage performance on cloud side simultaneously.

Key words: threshold deduplication, re-encryption, side channel attack, late verify

1 引言

随着大数据时代的到来，手机、平板电脑、穿戴式设备等便携式终端每天产生大量的用户数据，微型计算机、服务器等企业终端也不断生成海量的企业数据。为了缓解本地存储压力，越来越多的个人和企业用户选择将数据存储在云端[1]。在当前数据大爆炸的背景下，由于高冗余的特性，即使数据

收稿日期：2019−12−24；**修回日期**：2020−03−23

基金项目：国际关系学院国家安全高精尖学科建设科研专项基金资助项目（No.2019GA36）；国家自然科学基金资助项目（No.U1536207）；国家重点研发计划基金资助项目（No.2016QY04W0803）

Foundation Items: Special Funded by Construction of Advanced Disciplines for University of International Relations (No.2019GA36), The National Natural Science Foundation of China (No.U1536207), The National Key Research and Development Program of China (No.2016QY04W0803)

被外发，云存储按需付费的服务模式仍然要求用户付出大量不必要的开销，云服务提供商也面临存储管理重复数据的挑战。用户最关注的是其外发数据的安全性问题，即数据发送到云端存储后，是否会被云服务提供商和其他云用户窃取隐私。为了保护隐私，用户往往会选择将数据加密后再外发到云端存储。据统计，云端保存的密文数据中的 70%对应相同的明文[2]，而由于密文的不一致性，云服务提供商又无法使用简单的比对去重方法去除冗余数据。

加密云数据去重是解决以上问题的一种有效技术手段，其原理是让数据所有者基于数据本身生成一个内容相关密钥。例如，以数据的哈希值作为密钥[3]，然后使用该密钥对明文数据执行对称加密。这样，对相同的明文数据来说，其所有者可以生成相同的内容相关密钥，从而加密数据得到相同的密文。当密文上传到云端后，云端就可以执行去重。尽管这种方法有效地实现了密文去重，然而却存在边信道攻击[4]的风险。攻击者可以通过发起离线字典攻击，对于模板化生成的云端数据猜测其对应的明文内容，然后生成其内容相关密钥并加密，使用得到的密文测试去重系统，如果云端返回该密文数据已存在，则攻击者可以确认自己猜测的明文内容是正确的，从而窃取了云端数据的存在性隐私。为了应对边信道攻击，有研究者提出在密文生成过程中加入由第三方可信服务器生成的随机信息[2,4-7]。但是，攻击者仍然可以伪装成正常用户获取此随机信息以生成密文，在这种情况下，边信道攻击依然存在。

为了解决这一问题，国内外开展了大量的研究工作[8-14]，已有工作分别采用附加冗余流量和设置随机去重阈值的方式来混淆攻击者，使其难以判断所检测云数据的真实存在性。然而，第一类工作要求用户上传大量冗余数据[8]或要求云端维护额外的数据结构[9-10]，该类工作或无法实现云端数据存在性隐私的绝对安全，或即使实现安全，也需要云端付出大量的存储和计算开销。第二类工作通过设置随机阈值[11-13]，当同一文件的云端副本数达到阈值才执行去重，即使攻击者检测到数据被去重，也无法判断目标数据的归属。但是，这种方式的早期工作依赖于独立的第三方服务器[11]，存在单点失效的问题。近年来最新的研究进展是引入二次加密机制和 k 匿名隐私策略[12-13]，首先要求用户对使用内容相关密钥加密得到的密文分割并执行二次加密，由云端收集 k 个二次加密密文后，解密得到内容相关密钥加密的密文，再执行去重。在这种机制下，密文分割、二次加密密文生成和验证信息的计算需要用户付出大量的计算开销。并且，用户只能通过下载和解密云端生成的一次加密密文才可获得明文，而无法通过自己生成的二次加密密文解密得到明文。因此，为了在云端副本数未达到阈值时仍然可以下载密文并解密，用户还需独立上传私钥加密的密文，这给用户带来了大量额外的计算和通信开销。此外，云服务提供商为了验证 k 个副本的二次加密密文和一次加密密文的对应性，每一个副本都需用户分块生成验证信息，这将不可避免地带来较大的用户端计算开销。

为了解决这些问题，本文将探讨如何实现一种轻量级抗边信道攻击阈值去重方法。本文所提机制不仅能够实现去重的安全性，防止隐私泄露，而且需要确保去重方法的高效性，尤其是减少用户端的开销。特别地，本文提出并引入一种新型的阈值重加密机制，将用户端的密文分割转换为密钥分割，使用户只需基于内容相关密钥加密，云端就可基于密文重加密生成转换密文实现去重。并且，用户可从一次加密密文中直接解密出明文，从而避免了对明文数据多次加密和上传的开销。此外，所提机制还提供云端验证功能，使用户对同一文件只需生成一次验证标签，云服务提供商在对该文件的 k 个用户加密密文副本执行重加密后，可验证重加密密文与用户端明文的对应关系，从而确保去重方法的安全性。本文主要贡献如下。

1) 本文提出了一种支持轻量级云数据安全去重的系统模型，该模型考虑到用户端的性能特点，支持低用户开销的抗边信道攻击安全去重方法。

2) 在所提模型的基础上，本文提出了一种新型阈值重加密机制，该机制确保基于同一明文数据的 k 个不同密文副本可生成该明文数据的重加密密文。并且通过任何一个密文副本（含重加密密文），用户均可解密出明文。因为这一特性，用户只需基于内容相关密钥对明文数据执行加密，就可以由云端实现密文转换进而去重，并可以直接下载密文解密。

3) 基于所提的阈值重加密机制，本文提出了一个抗边信道攻击的轻量级去重协议。该协议规定由云端对同一明文数据的密文副本执行重加密和去重，确保即使攻击者检测到去重的发生，也无法判

断目标文件的归属，从而保护了云端数据的隐私安全。此外，所提协议还支持云服务提供商和用户端双向的数据完整性验证，确保数据的真实性和可用性。

4) 本文通过安全性分析证明了所提机制的安全性，并通过实验验证了其性能。结果表明，所提机制不仅是安全的，而且相比该领域的最新工作，性能有所提高，所需开销更低。

2 相关工作

在云数据的跨用户去重中，若云端存放了模板化的可预测文件，攻击者可以通过发起边信道攻击、上传猜测的目标文件并观察去重响应，从而窃取目标文件的存在性隐私。Armknecht 等 [15]分析了边信道攻击下隐私泄露的风险和云端去重效率之间的关系，发现通过降低去重效率可在一定程度上实现对该攻击的抵抗。因此，Zuo 等[8]提出了一种数据块级的安全去重方法，假设目标文件的所有敏感信息均存储在一个数据块中，其余块为公开块。当云端判断出所检测文件的真实存在性后，在命中文件和未命中文件的响应中分别添加不同数量的附加块，使 2 种情况下要求检测者上传的块数一样，从而确保检测者无法判断所检测文件的真实存在性。然而，在附加块攻击场景下，攻击者给检测文件附加随机数量的非命中块，再上传至去重系统检测。由于云端并不知道附加的非命中块数量，因此无法判断文件的真实存在性。此时，云服务提供商总是在相同数量范围中生成附加的冗余块，当冗余块数达到取值范围的边界点时，攻击者有可能窃取数据的存在性隐私。Yu 等[7]提出了一种数据块对（即 2 个数据块）去重检测方法，该方法能同时检测一个数据块对的存在性，并根据去重结果生成模糊化的响应。该方法可确保攻击者无法窃取数据存在性隐私，但需要云服务提供商维护一个独立的数据结构以记录所有用户检测过却未上传的数据块，加大了云端的存储和计算开销。基于该工作，Pooranian 等[9]在增加少量开销的基础上，实现了更强的云端数据隐私保护，然而，他们的工作仍然依赖于云端维护的数据结构。

作为一种新的途径，Harnik 等[11]通过设置随机阈值的方式去混淆攻击者，由服务器为每个文件独立地生成并保存一个阈值，当文件的存储副本数量达到阈值时，再执行云端去重。在这种方式下，即使攻击者通过去重检测，确定所检测的文件发生去重，也只能判断出目标文件的存在性。由于云端存放了多个目标文件的副本，这些副本可能来自不同的用户上传，故攻击者仍然无法确定该目标文件的归属关系。然而，这个方法的缺陷是需要依赖一个可信服务器去存储和维护每个文件的副本数量阈值[13]，这在现实中往往是难以保证的。Stanek 等[12,14]进一步将阈值去重拓展到加密数据的场景，以在抗边信道攻击的同时实现数据机密性的保护，他们在阈值去重的基础上，提出了“数据流行度”的概念，认为流行度低的文件需要更强的数据保护机制，反之，流行度高的文件可执行去重。然而，他们的工作仍然依赖于独立的第三方服务器为云端文件的副本计数。Zhang 等[13]在 Stanek 等工作的基础上，引入 k 匿名策略，实现了不依赖于第三方服务器的加密云数据阈值去重，要求用户对明文数据首先生成内容相关密钥，加密产生一次加密密文，然后在此基础上分割密文，通过多项式计算生成二次加密密文。对同一明文数据来说，一旦云端收集到 k 个不同的二次加密密文，其就可根据秘密分享机制的原理，恢复出相同的一次加密密文，不仅确保了数据的机密性，而且可实现云端数据去重。因此，当至少有 k 个用户上传了相同数据的二次加密密文后，用户端再次下载时将得到云端恢复出的一次加密密文，此时，用户只需使用明文数据的内容相关密钥即可实现解密。然而，由二次加密的过程可知，用户为了实现密文分割和二次加密，需要付出大量的计算开销。用户无法从二次加密密文中解密出一次加密密文，更无法直接解密出明文。为了确保少于 k 个用户上传同一数据的场景下用户仍然可实现下载解密，该方案要求用户同时为明文数据生成并上传一个用私钥加密的密文，这就给用户端带来了大量的额外开销。此外，云服务提供商为了验证 k 个副本的二次加密密文和一次加密密文的对应性，每一个副本都需用户分块生成验证信息，这将不可避免地带来较大的用户端计算开销。

此外，还有一些工作致力于加密云数据的安全去重。其中，Bellare 等[16]首次提出了 DupLESS 机制，通过在基于内容相关密钥加密明文的过程中引入随机度，使攻击者难以通过正常加密猜测的明文窃取云端密文的存在性隐私，从而实现了语义级的安全。在 Bellare 等工作的基础上，Liu 等[17]引入代理重签名技术，实现了一种支持标签去重的验证机制，进一步提高了去重的效率。Tang 等[18]基于重加

密技术和非交互式安全内容相关密钥生成技术设计了一种云数据安全去重方法，在实现相同等级安全性的前提下，进一步降低了用户端的开销。Dang 等[19]利用可信处理器在加密过程中实现随机度的引入。然而，这些工作均依赖在密文生成过程中引入随机度来实现对边信道攻击的抵抗。如果攻击者通过发起诸如女巫攻击[20]的手段，来伪装成注册用户，其同样可获得可信服务器产生的随机信息，从而实现窃取云端副本存在性隐私的目的。

3　准备工作

本节介绍本文所提方法对应的系统模型，并通过定义威胁模型来分析安全风险。为了应对风险，简要介绍了一些技术手段，如 CE 加密技术、秘密分享技术、重加密技术等，基于这些技术手段可构建所提的基于阈值重加密的抗边信道攻击云数据安全去重方法。

3.1　系统模型

本文考虑的系统模型由云服务提供商和用户这两类实体共同组成。一个云服务提供商为多个注册用户提供云存储及跨用户去重服务，其不仅提供按需付费的存储服务，而且具备强大的计算能力，能够在云端对用户数据实行计算并将结果返回。对于用户来说，考虑到数据的安全性，其应避免上传明文数据。在一次数据上传流程中，用户首先生成并上传数据的标签信息，云服务提供商根据标签在本地检索数据，以确定数据的存在性及副本数量。如果副本数量小于指定阈值，则通知用户上传密文数据及密文数据标签，后者用于实现对上传密文的完整性验证，同时对该副本继续计数。如果副本数量大于或等于阈值，则通知用户上传证明信息并基于所有权认证技术（POW, proof-of-ownership）验证用户对该数据的所有权，如果验证通过，则将相应用户加入该数据的所有权列表中，同时对该数据执行用户端去重，并继续计数。为了实现阈值去重并使攻击者无法根据去重响应的结果判断云端数据的真实归属关系，系统需要预先定义一个阈值 k，并同时对云服务提供商和用户公开。一旦相同明文对应的密文副本数量达到或超过 k，立即由云端对 k 个密文副本实现密文转换，使相同明文对应相同的密文，从而执行去重。值得说明的是，每个用户为自己的明文数据生成的密钥均包含全局唯一的用户标签信息以及原始数据信息，因此即使是相同的明文，不同用户将产生不同的密文。

3.2　威胁模型

在本文考虑的场景中，假设云服务提供商不完全可信。在大部分时间里，其能够诚实地为用户提供数据存储和去重服务，然而，其始终想要获得用户的数据内容信息。如果用户上传密文数据，其数据隐私将面临被云服务提供商窃取的风险。即使每个用户上传的都是密文数据，对同一明文数据来说，云服务提供商也会尝试基于来自不同用户的多个密文解密出明文。此外，为了节省存储空间，云服务提供商在数据副本数量没达到阈值的情况下，也会尝试丢弃用户的一些数据，并在用户下载时用其他的数据或伪造的数据代替用户数据返回。

从用户的角度来看，一方面，其有动机通过发起边信道攻击获取其他用户上传到云端的数据隐私信息；另一方面，其也会利用跨用户去重机制的特性，通过伪造虚假的证明信息，欺骗云服务提供商错误地将云端数据的所有权授予自己。在第一种情况下，用户按照模板格式生成其他用户的文件，猜测并填入敏感信息，然后按照协议生成标签及验证信息执行跨用户去重检测。如果云服务提供商提示数据在云端存在，则该用户可确认生成的文件即为目标用户所有。这种攻击方法对于按照模板格式生成的可预测用户文件隐私具有极大威胁。在第二种情况下，用户并没有目标文件，之前也没有上传过该文件，其猜测或通过某种渠道获得了目标文件的标签信息后欺骗云服务提供商，使自己通过所有权验证流程，从而非法获得云端数据的所有权。

3.3　CE 加密技术

CE 加密（convergent encryption）[21]是为了实现云数据的跨用户去重而设计的一种加密方法。该方法基于数据内容本身生成加密密钥，采用对称加密的方式实现加密，故拥有相同数据的不同用户不用协商即可生成相同密文，从而为实现跨用户去重提供支撑。从实现上来看，一个 CE 加密机制包含密钥生成、CE 加密和 CE 解密 3 个组成部分，以元算法的形式可表示如下。

$\text{cek} \leftarrow \text{CEKGen}(F,1^{\lambda})$。该算法以明文文件 F 和安全参数 λ 为输入，输出只和明文文件 F 有关的 CE 密钥 cek。其中，$F \in \{0,1\}^{*}$。一种简单的密钥生成方法为计算 F 的哈希值作为 cek 输出。

$c \leftarrow \text{CEEnc}(F,\text{cek})$。该算法以明文文件 F 和生成的 CE 密钥 cek 作为输入，对称加密后输出密文

c。由于在加密的过程中没有引入其他信息，CE 密钥 cek 只和明文文件 F 有关，因此密文 c 只和 F 有关。

$F \leftarrow \mathrm{CEDec}(c,\mathrm{cek})$。该算法以 CE 密文 c 和 CE 密钥 cek 作为输入，输出解密得到明文 F。由于 CE 加密机制采用对称加密，故解密密钥和加密密钥一样，均为 CE 密钥 cek。

3.4 秘密分享技术

秘密分享技术[22]是一种实现机密信息分享的轻量级技术，自提出以来，其被广泛地应用在分布式存储领域，以确保机密数据存储和分享的安全性。为了实现秘密分享，用户首先生成一个 k 次的多项式 $q(x)=a_0+a_1x+\cdots+a_{k-1}x^{k-1}$，将机密信息 D 作为零次项系数 a_0，然后随机生成 $k-1$ 个系数 $a_1,a_2,\cdots,a_{k-1}$，分别作为一次项到 $k-1$ 次项的系数。用户计算 $D_1=q(1),D_2=q(2),\cdots,D_n=q(n),n\geqslant k$。只有当在这 n 个多项式值中收集到不相同的任意 k 个值时，才能求解出 $k-1$ 次多项式，从而得到系数 $a_0,a_1,\cdots,a_{k-1}$，恢复出机密信息 D。在云存储场景中，假设有 n 个用户拥有机密信息 D 和一组系数 $a_1,a_2,\cdots,a_{k-1}$，每个用户根据自己的唯一标识 ID 计算出 $q(\mathrm{ID})$，部分用户会将计算出的多项式值上传到云端。云端只有收集到 k 个多项式值时，才能通过求解多项式得到机密信息 D，在其他情况下，无法获得用户的机密信息。

3.5 重加密技术

重加密技术[23]是一种通过第三方对用户加密的密文进行转换，从而允许密文的接收者使用新的密钥进行解密的技术。为了实现重加密技术，用户通常需要在生成加密密钥的同时生成转换密钥 rekey。用户使用加密密钥加密数据，将密文和转换密钥一同上传到云端并从本地删除。云端使用转换密钥对密文进行二次加密，保存密文并在接收到合法请求时发放给用户。从实现上看，一个重加密机制包含用户密钥生成、转换密钥生成、加密、重加密和解密这 5 个部分，可以用元算法形式表现如下。

$(\mathrm{pk},\mathrm{sk})\leftarrow \mathrm{KEYGen}(1^{\lambda})$。该算法以安全参数 λ 为输入，输出用户的公钥 pk 和私钥 sk。对于用户 i 来说，该算法输出其公私钥对 $(\mathrm{pk}_i,\mathrm{sk}_i)$。

$\mathrm{rk}_{ij}\leftarrow \mathrm{REKEYGen}(\mathrm{sk}_i,\mathrm{pk}_j)$。该算法以用户 i 的私钥 sk_i 和用户 j 的公钥 pk_j 为输入，输出重加密密钥 rk_{ij}。rk_{ij} 可将用户 i 加密的密文转换为用户 j 可解密的密文。

$C_i\leftarrow \mathrm{ENCRYPT}(F,\mathrm{pk}_i)$。该算法以明文文件 F 和用户 i 的公钥 pk_i 为输入，输出用户 i 的密文 C_i。

$C_j\leftarrow \mathrm{REENCRYPT}(C_i,\mathrm{rk}_{ij})$。该算法以用户 i 加密得到的密文 C_i 和用户 i 计算得到的重加密密钥 rk_{ij} 作为输入，输出重加密得到的密文 C_j。C_j 只能被用户 j 解密，因为其拥有 j 的私钥 sk_j。

$F\leftarrow \mathrm{DECRYPT}(C_j,\mathrm{sk}_j)$。该算法以密文 C_j 和用户 j 的私钥 sk_j 为输入，以解密出的明文文件 F 为输出。值得注意的是，C_j 可以是用户 j 使用公钥 pk_j 加密得到的密文，也可以是云服务提供商或其他第三方代理使用重加密密钥 rk_{ij} 对密文 C_i 重加密得到的密文。

4 基于阈值重加密的云数据去重方法

本节首先基于以上 CE 加密技术、秘密分享技术、重加密技术，构建阈值重加密基本算法，然后基于所构建算法设计一种抗边信道攻击的安全去重协议。所提协议保证恶意用户无法通过边信道攻击窃取指定用户的存在性隐私，同时协议的设计确保了开销可控。

4.1 阈值重加密基本算法

$(\mathrm{pk},\mathrm{sk})\leftarrow \mathrm{UKEYGen}(1^{\lambda})$。用户密钥生成算法输入安全参数 λ，输出用户的公私钥 pk 和 sk，分别用于数据标签的验证和签名。

$\mathrm{cek}\leftarrow \mathrm{CEKEYGen}(F,1^{\lambda})$。数据内容密钥生成算法输入明文文件 F 和安全参数 λ，输出密钥 cek。cek 是一种内容相关密钥。

$(K_1,K_2,\cdots,K_k)\leftarrow \mathrm{CEKDesp}(\mathrm{cek},k)$。数据内容加密密钥分解算法输入内容相关密钥 cek 和分解的份数 k，输出分解得到的 k 个密钥组件 $K_1,K_2,\cdots,K_k$。

$(C(\mathrm{UID}'),N_{\mathrm{UID}\_F})\leftarrow \mathrm{CEEnc}(F,\mathrm{cek},\mathrm{UID}',K_1,K_2,\cdots,K_k)$。数据内容加密算法的执行和用户相关，算法输入明文文件 F、内容相关密钥 cek、执行加密的用户唯一标识号 UID 对应的加密哈希值 UID' 和 k 个与数据内容相关的密钥组件 $K_1,K_2,\cdots,K_k$，输出该用户对应的密文信息 $C(\mathrm{UID}')$ 和辅助信息 N_{UID_F}。

$(\mathrm{tag}_1,N_{\mathrm{QF}},\mathrm{tag}_2,N_{\mathrm{VF}})\leftarrow \mathrm{TAGGen}(F,\mathrm{cek},sk_{\mathrm{UID}},K_1,K_2,\cdots,K_k)$。数据标签生成算法的执行和用户相

关，算法输入明文文件 F、内容相关密钥 cek 、用户 UID 的私钥 $\mathrm{sk}_{\mathrm{UID}}$，以及 k 个与数据内容相关的密钥组件 $K_1,K_2,\cdots,K_k$，输出数据查询信息 $(\mathrm{tag}_1,N_{\mathrm{QF}})$ 和数据所有权验证信息 $(\mathrm{tag}_2,N_{\mathrm{VF}})$。其中，前者用于查询 F 对应的密文在云端的存在性以及副本数量，后者用于在云端文件 F 的密文副本数量达到或超过阈值时，验证新请求上传用户 UID 对 F 的数据所有权。

$C_F \leftarrow \mathrm{REEncrypt}(C(\mathrm{UID}_1'),C(\mathrm{UID}_2'),\cdots,C(\mathrm{UID}_K'),N_{\mathrm{UID}_1\_F},N_{\mathrm{UID}_2\_F},\cdots,N_{\mathrm{UID}_k\_F})$。密文重加密算法由云端在收集得到 k 个不同用户对明文文件 F 生成的密文 $C(\mathrm{UID}_1'),C(\mathrm{UID}_2'),\cdots,C(\mathrm{UID}_K')$ 及辅助信息 $N_{\mathrm{UID}_1\_F},N_{\mathrm{UID}_2\_F},\cdots,N_{\mathrm{UID}_k\_F}$ 后，对密文执行重加密，从而产生统一的密文 C_F 用于去重。值得一提的是，本文所提的密文重加密算法不需要重加密密钥，且重加密后得到的密文 C_F 只需内容相关密钥 cek 即可解密。

$\{\mathrm{TRUE},\mathrm{FALSE}\} \leftarrow \mathrm{LATEVerify}(\mathrm{tag}_1,N_{\mathrm{QF}},F_C g^{\mathrm{cek}_{F_1}},g^{\mathrm{cek}_{F_2}},\cdots,g^{\mathrm{cek}_{F_k}})$。后验证算法由云端在计算得到 k 个重加密组件 $F_C g^{\mathrm{cek}_{F_1}},g^{\mathrm{cek}_{F_2}},\cdots,g^{\mathrm{cek}_{F_k}}$ 后，验证相应的 k 个用户上传数据前使用的查询信息 $(\mathrm{tag}_1,N_{\mathrm{QF}})$ 的正确性，并返回验证结果 TRUE 或 FALSE 。如果验证通过，说明这 k 个用户中的任一用户 $C(\mathrm{UID}_i'),i\in[1,k]$ 在生成密文 $C(\mathrm{UID}_i')$ 时使用的密钥组件与生成查询标签 tag_1 时使用的密钥组件是一致的，k 个密文 $C(\mathrm{UID}_1'),C(\mathrm{UID}_2'),\cdots,C(\mathrm{UID}_K')$ 及辅助信息 $N_{\mathrm{UID}_1\_F},N_{\mathrm{UID}_2\_F},\cdots,N_{\mathrm{UID}_k\_F}$ 均对应查询信息 $(\mathrm{tag}_1,N_{\mathrm{QF}})$ 标识的明文文件 F，因此，重加密密文 C_F 可用。如果验证不通过，说明这 k 个密文与查询信息对应的明文不一致，这 k 个密文中必然有一个或多个不对应明文文件 F，在这种情况下，重加密密文 C_F 不可用。

$\{\mathrm{TRUE},\mathrm{FALSE}\} \leftarrow \mathrm{POWVerify}(\mathrm{tag}_2,\mathrm{pk},C_F)$。所有权验证算法在同一明文文件 F 的云端密文副本数量达到阈值时由云端执行，其输入为云端要求文件 F 的后续请求上传者上传的数据验证标签 tag_2、用户公钥 pk 和此前已验证的 F 的真实重加密密文 C_F，算法输出所有权验证结果 TRUE 或 FALSE，分别标识公钥 pk 对应的用户对云端保存文件 F 的重加密密文 C_F 拥有所有权和没有所有权。

$F \leftarrow \mathrm{DECrypt}(C_F,\mathrm{cek})$。解密算法以重加密后得到的密文 C_F 和数据内容密钥 cek 作为输入，由用户执行解密，得到明文 F 作为输出。值得注意的是，解密过程不仅直接用到 cek ，而且用到由 cek 生成的解密密钥。

4.2　基于阈值重加密的抗边信道攻击安全去重协议

本节基于阈值重加密基本算法构建抗边信道攻击的安全去重协议。协议的构建分为以下 3 个阶段：用户上传阶段、CSP 去重及验证阶段和用户下载阶段。下面分别详细介绍每一阶段的协议流程，以展现完整的基于阈值重加密的抗边信道攻击安全去重方法。

4.2.1　用户上传阶段

用户上传阶段，用户需要执行 5 个元算法：UKEYGen 、CEKEYGen 、CEKDesp 、CEEnc 和 TAGGen ，分别对应用户密钥生成、数据内容密钥生成、密钥分割、数据加密和标签生成等步骤。考虑用户 i 想要将文件 F 上传到云端保存，其首先执行 UKEYGen 算法产生自己的公私钥对，然后执行 CEKEYGen 生成文件 F 的内容密钥，执行 CEKDesp 分割内容密钥，再通过 CEEnc 产生密文，通过 TAGGen 生成查询标签和验证标签。

用户密钥生成。令 G 和 G_T 是 2 个素数阶 p 的乘法循环群，$e:G\times G\rightarrow G_T$ 为双线性映射，g 为 G 的生成元。用户执行算法 UKEYGen ，随机选取 $x\in Z_p$ 且 $x\geqslant 2$ 作为自己的私钥 sk ，并计算对应的公钥 $\mathrm{pk}=g^x\in G$。

数据内容密钥生成。对于文件 F，用户执行算法 CEKEYGen 计算其内容相关密钥 $\mathrm{cek}_F=h(F)$，其中 $h(\cdot)$ 是选取的加密哈希函数。

密钥分割。对于文件 F 而言，用户执行算法 CEKDesp 将其内容相关密钥 cek_F 分解成 k 个密钥组件，记该密钥的长度为 L，其每个组件的最低 m 个比特位记为 $(\mathrm{cek}_{F_1},\mathrm{cek}_{F_2},\cdots,\mathrm{cek}_{F_k})$。显然，分割得到的 k 个内容相关密钥片段共同组成完整的内容相关密钥，用户如果拥有内容相关密钥 cek ，且知道分割的片段数量 k 以及选取的最低 m 个比特位，其一定可以得到 $(\mathrm{cek}_{F_1},\mathrm{cek}_{F_2},\cdots,\mathrm{cek}_{F_k})$。

数据加密。数据加密和用户相关，即使对同一文件 F，不同用户加密得到的密文不同，因为在本文考虑的场景中，每个用户均有一个全局唯一的标识 UID ，其取值是不为 1 的正整数，以 UID 为加密哈希函数的输入参数，计算 $\mathrm{UID}'=h(\mathrm{UID})$，与 cek_F 类似，UID' 的长度为 L。用户在加密数据时加入自

己的 UID 对应的信息，具体地，将 UID' 分成 $k-1$ 段，每一段取最低 m 个比特位，用 $(\mathrm{UID}')_i,(i\in[1,k-1])$ 表示。对于文件 F，用户执行算法 CEEnc 加密得到密文。由于每个用户的标识 UID 是唯一的，因此其对应的 UID' 也是唯一的，不会出现碰撞现象。按照以上的定义，假设未知数 x 可分为 $k-1$ 段，每一段取最低 m 个比特位，用 $x_i,(i\in[1,k-1])$ 表示。为了实现数据加密，首先定义多项式为

$$f_1(x)=\mathrm{cek}_{F_1}+\mathrm{cek}_{F_2}x_1+\mathrm{cek}_{F_3}x_2+\cdots+\mathrm{cek}_{F_k}x_{k-1} \quad (1)$$

其中，文件 F 对应的 k 个内容相关密钥片段分别作为 k 个多项式系数，多项式参数 x_i 表示用户标识信息。选取一个素数 q，对任一 x，定义辅助信息 $N=\dfrac{f_1(x)}{q}$，取模后的多项式值 $f(x)=f_1(x)\bmod q\in Z_q$。因此每个用户基于自己的标识和所拥有文件的内容相关密钥，均可计算得到一个多项式值。对文件 F 而言，如果其拥有者的标识信息为 UID，则文件所有者可计算出多项式值，如式(2)所示。

$$f_1(\mathrm{UID}')=\mathrm{cek}_{F_1}+\mathrm{cek}_{F_2}(\mathrm{UID}')_1+\mathrm{cek}_{F_3}(\mathrm{UID}')_2+\cdots+\mathrm{cek}_{F_k}(\mathrm{UID}')_{k-1} \quad (2)$$

从而计算出 $N_{\mathrm{UID}\_F}=\dfrac{f_1(\mathrm{UID}')}{q}$ 和 $f(\mathrm{UID}')=f_1(\mathrm{UID}')\bmod q\in Z_q$。显然，对文件 F 而言，只有得到其 k 个不同所有者产生的多项式值，才可通过求解方程组获得完整的内容相关密钥片段 $(\mathrm{cek}_{F_1},\mathrm{cek}_{F_2},\cdots,\mathrm{cek}_{F_k})$。在其他情况下，即使拥有 $k-1$ 个多项式值，也无法求解出任何一个内容相关的密钥片段。用户首先以 cek_F 为密钥，生成 F 的对称加密密文 F_C。用户可相应地计算出密文，如式(3)所示。

$$C(\mathrm{UID}')=F_C g^{f(\mathrm{UID}')} \quad (3)$$

标签生成。对文件 F 而言，由其所有者在上传之前执行算法 TAGGen 计算标签信息。首先，每个文件均由其所有者加密后上传到云端存储，由上文描述的加密流程可知，即使对相同文件，不同所有者加密产生的密文均不相同。而本文所提的机制要求对同一文件只有当云端存储的密文副本数量达到或超过阈值时才执行去重。因此，首先需要一种机制为相同文件的不同密文副本计数。为了实现这一目的，本文设计了一种数据查询标签的生成方法。查询标签一方面需要实现数据查询的功能，即无论云端存储的是同一数据的哪一个密文副本，通过查询标签都应准确地查询到该副本，并且能够验证查询标签和副本的对应关系，虽然这种验证可能会延后发生。另一方面，需要确保数据内容的安全性，即保证云服务提供商无法通过查询标签解密出数据的明文信息或内容相关密钥信息。因此，在此设计原则下，首先计算 1.5 对应的加密哈希值 $a=h(1.5)$，然后计算该值对应的 $f_1(a)$。根据式(2)可知，$f_1(a)$ 的取值范围无法确定，如果取值过大，则 $g^{f_1(a)}$ 计算开销会很大，从而难以定义数据查询标签。为了解决这一问题，本文定义 N_{QF} 为数据查询辅助信息，该变量专门用于记录 $f_1(a)$ 除以 q 所得的整数倍数，以使 $f_1(a)$ 可转换成一个范围有限的值，确保该值成为 g 的指数次方后易于计算。同时，定义的数据查询辅助信息可用于后续云服务提供商的后验证。具体地，数据查询辅助信息定义为 $N_{\mathrm{QF}}=\dfrac{f_1(a)}{q}$，此外，根据前文定义，计算 $f(a)=f_1(a)\bmod q\in Z_q$，将文件 F 设计的数据查询标签 tag_1 定义为

$$\sigma_{\mathrm{QF}}=F_C g^{f(a)} \quad (4)$$

首先，约定 $a=h(1.5)$ 作为参数后，所有用户只要拥有正确的明文信息，都可按照本文方法生成 cek、完成密钥分割，并按照式(2)、式(4)生成查询标签。由于密文副本的计算中引入的 UID 均为整数索引值，不同于查询标签计算引入的参数 1.5，故 tag_1 不会和文件 F 的密文副本混淆。tag_1 和密文副本对应关系的验证需要等到云服务提供商收集到至少 k 个不同用户产生的密文副本后才可开展，这部分将在后文 LATEVerify 中介绍。

除此之外，标签生成阶段还需为每一个待上传文件 F 生成一个验证标签 tag_2，一方面，文件 F 的所有者在下载数据时可通过该标签来验证数据的完整性；另一方面，当云端存储的 F 密文副本数量达到或超过阈值时，云服务提供商可通过该标签来验证后续请求上传用户对 F 的所有权。首先计算 1 对应的加密哈希值 $b=h(1)$，然后再计算 $f_1(b)$。与上文数据查询标签的计算类似，根据式(2)可知，$f_1(b)$ 的取值范围无法确定，如果取值过大，则 $g^{f_1(b)}$ 计算开销会很大，从而难以定义数据验证标签。为了解决这一问题，本文定义 N_{VF} 为数据验证辅助信息，该变量专门记录 $f_1(b)$ 除以 q 所得的整数倍数，

以使 $f_1(b)$ 可转换成一个范围有限的值，确保该值成为 g 的指数次方再计算。与此同时，定义的数据验证辅助信息可用于后续云服务提供商对重加密密文的计算。具体地，数据验证辅助信息定义为 $N_{\mathrm{VF}}=\dfrac{f_1(b)}{q}$，此外，根据前文定义，计算 $f(b)=f_1(b)\bmod q\in Z_q$，将文件 F 的验证标签 $\mathrm{tag}_2\in G$ 定义为

$$\sigma_{\mathrm{VF}}=(F_C g^{f(b)})^{\mathrm{sk}} \tag{5}$$

tag_2 的设计考虑了文件 F 的密文副本在云端的重加密操作。按照本文的设计，当云端副本数量达到或超过阈值时，云服务提供商需要对密文副本执行重加密操作，然后保存重加密密文并实行跨用户去重。为了支持验证后续请求上传用户对 F 的所有权，tag_2 的设计直接和重加密密文匹配，这部分可参考后文 REEncrypt 算法。此外，考虑到云端密文副本数量在没有达到阈值情况下文件 F 的所有者对其副本的下载和验证，tag_2 同样支持所有者在拥有文件 F 内容相关密钥的基础上，生成验证信息验证所下载副本的完整性，这部分在后文 DECrypt 算法中详细描述。

至此，用户需要准备的待上传加密数据及其标签信息均准备完毕。在文件 F 上传之前，用户首先将数据查询信息 $(\mathrm{tag}_1,N_{\mathrm{QF}})$ 发送给云服务提供商，根据云端发回的响应决定是上传加密副本、辅助信息 $N_{\mathrm{UID\_}F}$ 和数据验证信息 $(\mathrm{tag}_2,N_{\mathrm{VF}})$，还是仅上传 $(\mathrm{tag}_2,N_{\mathrm{VF}})$ 及用户公钥证明自己的所有权。

4.2.2　CSP 去重及验证阶段

CSP 去重及验证阶段需要云服务提供商执行 3 个元算法：REEncrypt、LATEVerify 和 POWVerify，分别对应重加密、数据查询标签后验证和所有权验证 3 个过程。对同一文件 F，云端使用其数据查询标签 tag_1 进行标识，一旦计数达到或超过阈值，即执行 REEncrypt 对该文件对应的所有密文副本重加密，产生统一密文。同时，云服务提供商执行 LATEVerify 对文件 F 的数据查询信息 $(\mathrm{tag}_1,N_{\mathrm{QF}})$ 进行验证，确认 $(\mathrm{tag}_1,N_{\mathrm{QF}})$ 和用作重加密的每一个密文副本的对应关系是否正确。完成这两项工作后，云服务提供商对文件 F 参与重加密计算的密文副本开展跨用户去重，如果后续还有 F 的所有者请求上传，云服务提供商要求其上传数据验证信息 $(\mathrm{tag}_2,N_{\mathrm{VF}})$ 及公钥 pk，执行 POWVerify 验证其所有权。

重加密。对同一文件 F 而言，云服务提供商在本地使用其查询标签 tag_1 作为计数标识。一旦文件 F 的云端密文副本数量达到或超过阈值 k，则由云端对 tag_1 对应的所有密文副本执行算法 REEncrypt 开展重加密。需要注意的是，如果 F 不同的密文副本数量超过 k，则由云服务提供商随机抽取 k 个计算重加密密钥组件，组件全部计算完毕后再对所有密文副本执行重加密。为了简单起见，这里假设 F 对应的密文副本数量为 k，它们各自的上传者标识信息分别为 $\mathrm{UID}_1',\mathrm{UID}_2',\cdots,\mathrm{UID}_k'$，对云服务提供商公开。将这 k 个密文的加密过程展开，如式(6)所示。

$$\begin{cases} F_C g^{\mathrm{cek}_{F_1}+\mathrm{cek}_{F_2}(\mathrm{UID}_1')_1+\mathrm{cek}_{F_3}(\mathrm{UID}_1')_2+\cdots+\mathrm{cek}_{F_k}(\mathrm{UID}_1')_{k-1}}=C(\mathrm{UID}_1')g^{qN_{\mathrm{UID}_1\_F}} \\ F_C g^{\mathrm{cek}_{F_1}+\mathrm{cek}_{F_2}(\mathrm{UID}_2')_1+\mathrm{cek}_{F_3}(\mathrm{UID}_2')_2+\cdots+\mathrm{cek}_{F_k}(\mathrm{UID}_2')_{k-1}}=C(\mathrm{UID}_2')g^{qN_{\mathrm{UID}_2\_F}} \\ \vdots \\ F_C g^{\mathrm{cek}_{F_1}+\mathrm{cek}_{F_2}(\mathrm{UID}_k')_1+\mathrm{cek}_{F_3}(\mathrm{UID}_k')_2+\cdots+\mathrm{cek}_{F_k}(\mathrm{UID}_k')_{k-1}}=C(\mathrm{UID}_k')g^{qN_{\mathrm{UID}_k\_F}} \end{cases} \tag{6}$$

为了实现重加密，同时确保文件 F 的密钥组件 $\mathrm{cek}_{F_1},\mathrm{cek}_{F_2},\cdots,\mathrm{cek}_{F_k}$ 不泄露给云服务提供商，首先需要从式(6)所示方程组中求解出 k 个重加密组件 $F_C g^{\mathrm{cek}_{F_1}}$ 和 $g^{\mathrm{cek}_{F_2}},\cdots,g^{\mathrm{cek}_{F_k}}$。为了实现求解，首先将方程组从上到下相邻方程两两相除，则未知数 $F_C g^{\mathrm{cek}_{F_1}}$ 被消掉，可得

$$\begin{cases} g^{\mathrm{cek}_{F_2}((\mathrm{UID}_1')_1-(\mathrm{UID}_2')_1)+\mathrm{cek}_{F_3}((\mathrm{UID}_1')_2-(\mathrm{UID}_2')_2)+\cdots+\mathrm{cek}_{F_k}((\mathrm{UID}_1')_{k-1}-(\mathrm{UID}_2')_{k-1})}= \\ \dfrac{C(\mathrm{UID}_1')g^{qN_{\mathrm{UID}_1\_F}}}{C(\mathrm{UID}_2')g^{qN_{\mathrm{UID}_2\_F}}} \\ g^{\mathrm{cek}_{F_2}((\mathrm{UID}_2')_1-(\mathrm{UID}_3')_1)+\mathrm{cek}_{F_3}((\mathrm{UID}_2')_2-(\mathrm{UID}_3')_2)+\cdots+\mathrm{cek}_{F_k}((\mathrm{UID}_2')_{k-1}-(\mathrm{UID}_3')_{k-1})}= \\ \dfrac{C(\mathrm{UID}_2')g^{qN_{\mathrm{UID}_2\_F}}}{C(\mathrm{UID}_3')g^{qN_{\mathrm{UID}_3\_F}}} \\ \vdots \\ g^{\mathrm{cek}_{F_2}((\mathrm{UID}_{k-1}')_1-(\mathrm{UID}_k')_1)+\mathrm{cek}_{F_3}((\mathrm{UID}_{k-1}')_2-(\mathrm{UID}_k')_2)+\cdots+\mathrm{cek}_{F_k}((\mathrm{UID}_{k-1}')_{k-1}-(\mathrm{UID}_k')_{k-1})}= \\ \dfrac{C(\mathrm{UID}_{k-1}')g^{qN_{\mathrm{UID}_{k-1}\_F}}}{C(\mathrm{UID}_k')g^{qN_{\mathrm{UID}_k\_F}}} \end{cases} \tag{7}$$

式(7)包含 $k-1$ 个方程和 $k-1$ 个未知数 $g^{\mathrm{cek}_{F_2}}$，$g^{\mathrm{cek}_{F_3}},\cdots,g^{\mathrm{cek}_{F_k}}$。通过求解，不难得到 $g^{\mathrm{cek}_{F_2}},g^{\mathrm{cek}_{F_3}}\cdots,g^{\mathrm{cek}_{F_k}}$，将它们代入式(6)的任一方程，可得 $F_C g^{cek_{F_1}}$ 的值。此时，云服务提供商计算式(8)作为重加密密文。

$$C(b)=F_C g^{\mathrm{cek}_{F_1}} g^{\mathrm{cek}_{F_2}b_1} g^{\mathrm{cek}_{F_2}b_2}\cdots\cdots g^{\mathrm{cek}_{F_k}b_{k-1}} g^{-qN_{VF}} \tag{8}$$

值得注意的是，k 个副本对应的上传者标识均不为 1，所以通过以上方法重加密得到的密文 $C(1)$ 和 k 个密文副本均不相同。另外，假设 $k=40$，则 UID' 被分成 39 段，每一段取最低 4 个比特位。对攻击者来说，如果要得到 F_C，那么破解的概率为

$\left(\frac{1}{2^4}\right)^{39}$，量级为$10^{-46}$，所以在未收集齐 F 对应的 k 个密文副本情况下，成功实现重加密的概率极低。

数据查询标签后验证。当文件 F 在云端存储的密文副本数量达到或超过阈值时，即可由云服务提供商执行 LATEVerify 开展后验证，以检验文件 F 的数据查询信息$(\text{tag}_1, N_{\text{QF}})$和 k 个密文副本是否对应。在以上重加密的步骤中，当云服务提供商集齐一个文件的副本后，其首先可通过求解方程组得到 $k-1$ 个加密组件 $g^{\text{cek}_{F_2}}, g^{\text{cek}_{F_3}} \cdots, g^{\text{cek}_{F_k}}$ 以及 $F_C g^{\text{cek}_{F_1}}$。此时，云服务提供商可按照事先约定，以 $a=h(1.5)$ 为输入参数，计算

$$C(a)=F_C g^{\text{cek}_{F_1}} g^{\text{cek}_{F_2} a_1} g^{\text{cek}_{F_3} a_2} \cdots g^{\text{cek}_{F_k} a_{k-1}} g^{-qN_{\text{QF}}} \tag{9}$$

将式(9)计算得到的 $C(a)$ 和文件 F 对应的 tag_1 比较，如果一致，说明 k 个密文 $C(\text{UID}_1'), C(\text{UID}_2'), \cdots, C(\text{UID}_k')$ 均对应查询标签 tag_1 标识的明文文件 F，因此重加密密文 $C(b)$ 可用。如果 $C(a)$ 和 F 的 tag_1 不一致，说明存在文件 F 的密文上传者，其计算密文使用的密钥组件和计算数据查询标签使用的密钥组件不一致。在这种情况下，云服务提供商必须重新找到 k 个 F 的密文副本，重新计算重加密密文，直到使用计算出的重加密组件计算出 F 的数据查询标签和 tag_1 一致时，重加密密文才可被采纳使用。

所有权验证。当文件 F 在云端存储的密文副本数量达到或超过阈值，云服务提供商生成了正确的重加密密文且被采纳时，云服务提供商会要求 F 的后续请求上传者上传 F 的数据验证标签 tag_2 及用户公钥 pk。此时，云服务提供商基于 F 的重加密密文 $C(b) \in G$，验证式(10)所示的双线性等式是否成立。

$$e(C(b), \text{pk}) = e(\text{tag}_2, g) \tag{10}$$

如果式(10)成立，说明用户签名的 tag_2 是文件 F 正确的验证标签，根据式(5)的 tag_2 定义可知，计算 tag_2 的过程中需要用户使用 F 及其对应的内容密钥组件。式(10)验证通过，说明计算 tag_2 的用户使用的 F 与云端产生的重加密密文 $C(b)$ 对应的 k 个密文副本加密过程中使用的 F 是一致的。因此可证明用户对 F 的所有权。

4.2.3 用户下载阶段

用户下载阶段，用户只需执行算法 DECrypt 完成对所下载密文的解密及验证。对于文件 F 来说，用户的下载分为 2 种情况。第一种情况是在云端 F 的密文副本数量没有达到阈值时，用户下载 F 的密文副本及验证标签 tag_2。假设用户的标识信息为 UID，则此时用户下载的是自己加密的副本。考虑到云服务提供商对密文可能的损坏或篡改，将下载的密文副本记为 $C'(\text{UID}')$。第二种情况是在云端 F 的密文副本数量达到或超过阈值时，此时云服务提供商已经完成了对 F 密文副本的重加密和数据查询标签的后验证，如果验证通过，用户下载到的应是 F 的重加密密文 $C(b)$。在第一种情况下，用户首先基于自己保存的 F 的数据内容相关密钥片段 $(\text{cek}_{F_1}, \text{cek}_{F_2}, \cdots, \text{cek}_{F_k})$ 计算式(11)和式(12)。

$$g^{f(b)} = g^{(\text{cek}_{F_1}+\text{cek}_{F_2} b_1+\text{cek}_{F_3} b_2 \cdots+\text{cek}_{F_k} b_{k-1}) \bmod q} \tag{11}$$

$$g^{f(\text{UID}')} = g^{(\text{cek}_{F_1}+\text{cek}_{F_2}(\text{UID}')_1+\text{cek}_{F_3}(\text{UID}')_2+\cdots+\text{cek}_{F_k}(\text{UID}')_{k-1}) \bmod q} \tag{12}$$

其中，$f(b) \in Z_q, f(\text{UID}') \in Z_q$。然后，对下载的密文副本 $C'(\text{UID}')$ 重加密，产生新的密文，如式(13)所示。

$$C'(b) = \frac{C'(\text{UID}')}{g^{f(\text{UID}')}} g^{f(b)} \tag{13}$$

基于 tag_2 和 $C'(b)$，可通过计算式(14)所示的双线性等式是否成立，来验证所下载密文副本的正确性。

$$e(C'(b), \text{pk}) = e(\text{tag}_2, g) \tag{14}$$

如果式(14)成立，用户可通过式(15)恢复出正确的 F_C；否则，可证明下载的密文副本 $C'(\text{UID}')$ 并非自己上传的版本。

$$F_C = \frac{C'(\text{UID}')}{g^{f(\text{UID}')}} \tag{15}$$

基于文件 F 的内容相关密钥对 F_C 解密可得到明文 F。

在第二种情况下，用户按照式(16)计算所下载重加密密文 $C(b)$ 的正确性。

$$e(C(b), \text{pk}) = e(\text{tag}_2, g) \tag{16}$$

若通过，则用户基于式(11)计算 $g^{f(b)}$，并通过式(17)恢复正确 F_C；否则，说明下载的重加密密文 $C(b)$ 是错误的。

$$F_C = \frac{C(b)}{g^{f(b)}} \tag{17}$$

基于文件 F 的内容相关密钥对 F_C 解密可得到明文 F。

5 正确性验证及安全性分析

本节首先在正确性验证部分验证所提机制中核心算法的正确性，然后通过安全性分析来分析所提协议的安全风险。

5.1　正确性验证

定理 1　只要收集到文件 F 的任意 k 个不同的密文副本，云服务提供商通过执行 REEncrypt 算法总能正确地生成 F 的重加密密文。

证明　收集到文件 F 的任意 k 个不同密文副本后，云服务提供商可求解式(6)的方程组。方程组包含 k 个 $F_C g^{\mathrm{cek}_{F_1}}$ 方程，需要求解的未知数分别为 $F_C g^{\mathrm{cek}_{F_1}}$ 和 $g^{\mathrm{cek}_{F_2}}, g^{\mathrm{cek}_{F_2}}, \cdots, g^{\mathrm{cek}_{F_k}}$。将这 k 个未知数分别记为 x_1, $x_2, \cdots, x_k$，将 $C(\mathrm{UID}_1')g^{qN_{\mathrm{UID}_1\_F}}$，$C(\mathrm{UID}_2')g^{qN_{\mathrm{UID}_2\_F}}, \cdots$, $C(\mathrm{UID}_k')g^{qN_{\mathrm{UID}_k\_F}}$ 记为 $y_1, y_2, \cdots, y_k$。则式(6)的方程组可写成

$$\begin{cases} x_1 x_2^{(\mathrm{UID}_1')_1} x_3^{(\mathrm{UID}_1')_2} \cdots x_k^{(\mathrm{UID}_1')_{k-1}} = y_1 \\ x_1 x_2^{(\mathrm{UID}_2')_1} x_3^{(\mathrm{UID}_2')_2} \cdots x_k^{(\mathrm{UID}_2')_{k-1}} = y_2 \\ \vdots \\ x_1 x_2^{(\mathrm{UID}_k')_1} x_3^{(\mathrm{UID}_k')_2} \cdots x_k^{(\mathrm{UID}_k')_{k-1}} = y_k \end{cases} \tag{18}$$

对式(18)的 k 个方程参照式(7)进行求解，两两相除，两边以 g 为底取对数，可得

$$\begin{cases} ((\mathrm{UID}_1')_1 - (\mathrm{UID}_2')_1)\log x_2 + ((\mathrm{UID}_1')_2 - (\mathrm{UID}_2')_2 \log x_3 + \cdots + \\ ((\mathrm{UID}_1')_{k-1} - (\mathrm{UID}_2')_{k-1})\log x_k = \log \dfrac{y_1}{y_2} \\ ((\mathrm{UID}_2')_1 - (\mathrm{UID}_3')_1)\log x_2 + ((\mathrm{UID}_2')_2 - (\mathrm{UID}_3')_2 \log x_3 + \cdots + \\ ((\mathrm{UID}_2')_{k-1} - (\mathrm{UID}_3')_{k-1})\log x_k = \log \dfrac{y_2}{y_3} \\ \vdots \\ ((\mathrm{UID}_{k-1}')_1 - (\mathrm{UID}_k')_1)\log x_2 + ((\mathrm{UID}_{k-1}')_2 - (\mathrm{UID}_k')_2 \log x_3 + \cdots + \\ ((\mathrm{UID}_{k-1}')_{k-1} - (\mathrm{UID}_k')_{k-1})\log x_k = \log \dfrac{y_{k-1}}{y_k} \end{cases} \tag{19}$$

将式(19)中的 $\log x_i$ 记为 $x_i', i \in [2,k]$，将 $\log \dfrac{y_1}{y_2}$, $\log \dfrac{y_2}{y_3}$, …, $\log \dfrac{y_{k-1}}{y_k}$ 分别记为 $y_2', y_3', \cdots, y_k'$。则式(19)可写为

$$\begin{cases} ((\mathrm{UID}_1')_1 - (\mathrm{UID}_2')_1)x_2' + ((\mathrm{UID}_1')_2 - (\mathrm{UID}_2')_2 x_3' + \cdots + \\ ((\mathrm{UID}_1')_{k-1} - (\mathrm{UID}_2')_{k-1})x_k' = y_2' \\ ((\mathrm{UID}_2')_1 - (\mathrm{UID}_3')_1)x_2' + ((\mathrm{UID}_2')_2 - (\mathrm{UID}_3')_2 x_3' + \cdots + \\ ((\mathrm{UID}_2')_{k-1} - (\mathrm{UID}_3')_{k-1})x_k' = y_3' \\ \vdots \\ ((\mathrm{UID}_{k-1}')_1 - (\mathrm{UID}_k')_1)x_2' + ((\mathrm{UID}_{k-1}')_2 - (\mathrm{UID}_k')_2 x_3' + \cdots + \\ ((\mathrm{UID}_{k-1}')_{k-1} - (\mathrm{UID}_k')_{k-1})x_k' = y_k' \end{cases} \tag{20}$$

式(18)所示的方程组已转换为式(20)所示的 $k-1$ 元一次方程组，由于方程组包含 $k-1$ 个方程，故可按照 k 元一次方程组求解方法求解出 $x_2', x_3', \cdots, x_k'$。对于每一个 $x_i', i \in [2,k]$，可通过式(21)恢复 x_i。

$$x_i = g^{x_i'} \tag{21}$$

将求出的 $x_2, x_3, \cdots, x_k$ 代入式(18)的任一方程都可求出 x_1。至此，全部 k 个未知数 $F_C g^{\mathrm{cek}_{F_1}}$ 和 $g^{\mathrm{cek}_{F_2}}, g^{\mathrm{cek}_{F_2}}, \cdots, g^{\mathrm{cek}_{F_k}}$ 已解出。此后，云服务提供商可按照式(9)计算出 F 的数据查询标签 $C(a)$，并与以上求解方程所用的 k 个密文副本所对应的查询标签 tag_1 对比，如果不一致，说明对应的密文副本是错误的；否则，说明求解出 $g^{\mathrm{cek}_{F_2}}, g^{\mathrm{cek}_{F_2}}, \cdots, g^{\mathrm{cek}_{F_k}}$ 这 $k-1$ 个未知数均是计算文件 F 查询标签的内容相关密钥组件，同时 $F_C g^{\mathrm{cek}_{F_1}}$ 中也包含正确的密钥组件，基于它们，按照式(8)计算出的重加密密文则为文件 F 的正确重加密密文。

证毕。

5.2　安全性分析

定理 2　假设攻击者截获了用户 UID 对文件 F 的加密密文副本 $C(\mathrm{UID}')$、数据查询标签 tag_1 或云服务提供商重加密密文 $C(b)$，其无法解密出文件 F 的明文信息。

证明　由 $C(\mathrm{UID}')$、tag_1 和 $C(b)$ 的计算式（式(3)、式(4)和式(8)）可知，文件 F 的加密密文副本、数据查询标签及云服务提供商的重加密密文均为 $F_C g^{f(x)}$ 的形式，其中变量 x 在 3 种情况下分别对应 UID'、$h(1.5)$ 和 $h(1)$。对 $F_C g^{f(x)}$ 取对数可得

$$\log F_C g^{f(x)} = \log F_C + f(x)\log g \tag{22}$$

因此，攻击者要从 $F_C g^{f(x)}$ 中获取明文文件 F 的信息，等价于从 $\log F_C g^{f(x)}$ 中获取 $f(x)\log g$ 的信息，或从 $\log F_C$ 中获取明文 F 的信息。由于 $\log g$ 为公开信息，因此获取 $f(x)\log g$ 的信息等价于攻击者获取 $f(x)$ 的信息。由 $f(x)$ 的定义可知，该多项式的系数均为文件 F 的内容相关密钥组件。根据密钥生成和密钥分割步骤可知，只有拥有原文件，才能计算出其内容相关密钥，进而分割产生内容相关密钥组件。因而攻击者在截获 $C(\mathrm{UID}')$、tag_1 或 $C(b)$ 的情况下能够解密出文件 F 的明文信息等价于攻击者已经知道文件 F 的明文信息。另一方面，由本文定义可知，$f(x) \in z_q$，攻击者以 $\dfrac{1}{q-1}$ 的概率分别猜测

了 $f(a)$、$f(\mathrm{UID}')$、$f(b)$ 的取值后，计算 $g^{f(a)}$、$g^{f(\mathrm{UID}')}$、$g^{f(b)}$，用 $C(a)$、$C(\mathrm{UID}')$、$C(b)$ 分别除以猜测得到的 $g^{f(a)}$、$g^{f(\mathrm{UID}')}$ 和 $g^{f(b)}$，可以得到 F 的密文 F_C。对攻击者来说，其没有 F 的内容相关密钥 cek_F，故无法通过解密 F_C 得到明文 F。由于 cek_F 是基于 F 得到的，因此可以得到结论，攻击者只有在获取明文 F 的前提下才能解密 F_C 得到 F，形成矛盾。此外，由以上分析可知，攻击者从 $\log F_C$ 中获取明文 F 的信息，同样只有在获取明文 F 的前提下才可实现。综上，即使攻击者截获了用户 UID 对文件 F 的加密密文副本 $C(\mathrm{UID}')$、数据查询标签 tag_1 或云服务提供商重加密密文 $C(b)$，其仍然无法解密出文件 F 的明文信息。

证毕。

定理 3 假设攻击者获得的文件 F 的加密密文副本数量少于阈值 k，则其无法实现跨用户去重。

证明 根据本文所设计的去重方法，只有当云端保存的文件 F 的加密密文副本数量达到或超过阈值 k 时，云端才可执行重加密操作，将这些密文副本转换为统一的重加密密文，从而允许云服务提供商实行跨用户去重。而在攻击者获得的文件 F 的加密密文副本数量为 $\delta(\delta<k)$ 时，其若想实现跨用户去重，以窃取云端其他用户数据的存在性隐私，就需要能够从 δ 个密文副本中生成文件 F 的重加密密文，并通过用户端的验证。

假设攻击者所获取的 δ 个密文副本为 $C(\mathrm{UID}_1'),C(\mathrm{UID}_2'),\cdots,C(\mathrm{UID}_\delta')$，辅助信息为 $N_{\mathrm{UID}_1\_F},N_{\mathrm{UID}_2\_F},\cdots,N_{\mathrm{UID}_\delta\_F}$，其若要从它们中生成重加密密文，等价于求解式(23)所示的方程组。

$$\begin{cases} F_C g^{\mathrm{cek}_{F_1}+\mathrm{cek}_{F_2}(\mathrm{UID}_1')_1+\mathrm{cek}_{F_3}(\mathrm{UID}_1')_2+\cdots+\mathrm{cek}_{F_k}(\mathrm{UID}_1')_{k-1}}= \\ C(\mathrm{UID}_1')g^{qN_{\mathrm{UID}_1\_F}} \\ F_C g^{\mathrm{cek}_{F_1}+\mathrm{cek}_{F_2}(\mathrm{UID}_2')_1+\mathrm{cek}_{F_3}(\mathrm{UID}_2')_2+\cdots+\mathrm{cek}_{F_k}(\mathrm{UID}_2')_{k-1}}= \\ C(\mathrm{UID}_2')g^{qN_{\mathrm{UID}_2\_F}} \\ \vdots \\ F_C g^{\mathrm{cek}_{F_1}+\mathrm{cek}_{F_2}(\mathrm{UID}_\delta')_1+\mathrm{cek}_{F_3}(\mathrm{UID}_\delta')_2+\cdots+\mathrm{cek}_{F_k}(\mathrm{UID}_\delta')_{k-1}}= \\ C(\mathrm{UID}_\delta')g^{qN_{\mathrm{UID}_\delta\_F}} \end{cases} \tag{23}$$

参照式(7)计算方法，对式(23)方程组里的方程两两相除，消掉 $F_C g^{\mathrm{cek}_{F_1}}$，得到 $k-1$ 个方程，对两边取对数。参照式(18)~式(20)，将 $\log g^{\mathrm{cek}_{F_2}}$，$\log g^{\mathrm{cek}_{F_3}},\cdots,\log g^{\mathrm{cek}_{F_k}}$ 记为 $x_i',i\in[2,\delta-1]$。再将等式右边记为 $y_2',y_3'\cdots,y_\delta'$，则式(23)所示方程组可写成

$$\begin{cases} ((\mathrm{UID}_1')_1-(\mathrm{UID}_2')_1)x_2'+((\mathrm{UID}_1')_2-(\mathrm{UID}_2')_2)x_3'+\cdots+ \\ ((\mathrm{UID}_1')_{k-1}-(\mathrm{UID}_2')_{k-1})x_k'=y_2' \\ ((\mathrm{UID}_2')_1-(\mathrm{UID}_3')_1)x_2'+((\mathrm{UID}_2')_2-(\mathrm{UID}_3')_2)x_3'+\cdots+ \\ ((\mathrm{UID}_2')_{k-1}-(\mathrm{UID}_3')_{k-1})x_k'=y_3' \\ \vdots \\ ((\mathrm{UID}_{\delta-1}')_1-(\mathrm{UID}_\delta')_1)x_2'+((\mathrm{UID}_{\delta-1}')_2-(\mathrm{UID}_\delta')_2)x_3'+\cdots+ \\ ((\mathrm{UID}_{\delta-1}')_{k-1}-(\mathrm{UID}_\delta')_{k-1})x_k'=y_\delta' \end{cases} \tag{24}$$

攻击者要想生成 F 的重加密密文，其必须求解出 $g^{\mathrm{cek}_{F_2}},\cdots,g^{\mathrm{cek}_{F_k}}$ 这 $k-1$ 个内容相关密钥组件以及 $F_C g^{\mathrm{cek}_{F_1}}$ 这个含 F_C 信息的密钥组件。因此，在式(24)所示的由 $\delta-1$ 个方程组成的方程组中，需要求解出 $k-1$ 个未知数，再通过式(21)转换，得到重加密所需的 k 个组件。根据非齐次线性方程组解的存在性条件可知，由于 $\delta-1<k-1$，式(24)所示方程组的系数矩阵的秩小于 $k-1$，方程组有无穷多解。因此，攻击者无法求解出文件 F 的内容相关密钥对应的唯一密钥组件。所以其无法产生重加密密文以实现跨用户去重。

证毕。

6 实验验证及结果分析

本节通过实验来验证本文所提的基于阈值重加密的抗边信道攻击安全去重方法的性能，并与该领域最新成果对比，证明本文方法的轻量级特点。为了开展实验，本节基于 Gmpy2 2.0.8 和 PyCrypto 2.6.1，并采用 Python 来实现本文所提的基于阈值重加密的抗边信道攻击云数据安全去重方法。所构建的云数据去重系统部署在亚马逊 EC2（elastic computing cloud）上。在本文构建的系统中，哈希函数采用 SHA256 实现，比照对象的对称加密算法实现为 AES256。选取一组配置为 Intel(R) Core(TM) i5-8259U CPU @ 2.30 GHz、8 GB RAM 内存和 1TB 容量 7 200 转硬盘的服务器部署客户端，实现所提协议中客户端密钥生成、密文生成、标签生成以及验证等工作。本节所展示的实验结果均为 50 次实验所得平均值。

6.1 用户端计算开销随文件大小变化的关系

本节实验关注协议在执行过程中的用户端计算开销随文件大小变化的关系，并将本文所提方法对应的结果和文献[13]的工作对应开销进行比较，

从而验证本文所提方法的性能。为了保证比较条件的一致性，本节在使用相同测试文件的基础上，采用相同的 cek 生成方法、相同的公开和私密参数生成方法，并将 2 种去重方法的阈值均设置为相同数值。在此场景下，分别将阈值 k 设置为 15、35 和 60，选取的最低比特 m=4，在用户上传阶段比较本文所提方法和文献[13]方法对不同大小文件的用户端计算开销（为了在同等条件下比较，二者的查询标签信息计算开销均忽略不计），采用算法平均执行时间来衡量计算开销，结果如图 1 所示。

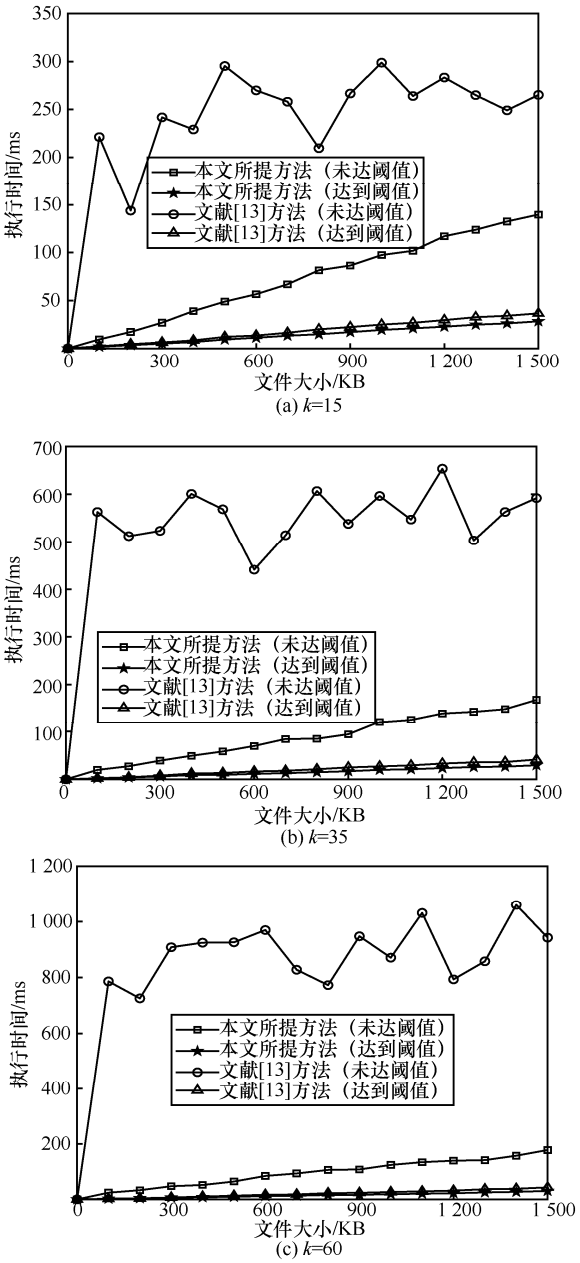


图 1　不同文件大小下算法平均执行时间比较

在图 1 中，对相同文件比较了其云端副本数量未达阈值、达到阈值情况下，所提方法和文献[13]方法的用户端计算时间随文件大小变化的关系。可以看出，当文件大小从 0 增加到 1 500 KB 时，4 条曲线对应的纵坐标取值均呈整体增长趋势。在云端副本数未达阈值的情况下，所提方法的执行时间显著低于文献[13]方法。这是由于文献[13]方法要求用户生成并上传二次加密的密文 $C(i)$、用户私钥加密的密文 $C'(i)$ 和辅助信息 aux，尤其是 aux 涉及大量的幂指数运算。而所提方法只需用户生成并上传密文 $C(\mathrm{UID})$ 和验证信息 $(\mathrm{tag}_2, N_{\mathrm{VF}})$。相比辅助信息 aux 的生成，$\mathrm{tag}_2$ 和 N_{VF} 生成所需的幂指数运算量显著降低。在云端副本数达到阈值的情况下，所提方法的开销略低于文献[13]方法。这是因为文献[13]方法要求用户分割一次加密密文、生成并上传二次加密的密文 $C(i)$，所提方法将密文分割转变为密钥分割，仅要求用户生成并上传用户公钥 pk 及 $(\mathrm{tag}_2, N_{\mathrm{VF}})$，在文件大小较小时，$\mathrm{tag}_2$ 计算中涉及的乘法运算量主要由 $g^{f(1)}$ 决定，由于取模后所得结果的不确定性，会出现一些轻微波动。综合比较，所提方法的平均执行时间显著低于文献[13]方法，且 2 个算法在云端副本数量达到或超过阈值后的用户计算开销均低于未超过阈值情况。

6.2　用户端计算开销随分块数量 k 变化的关系

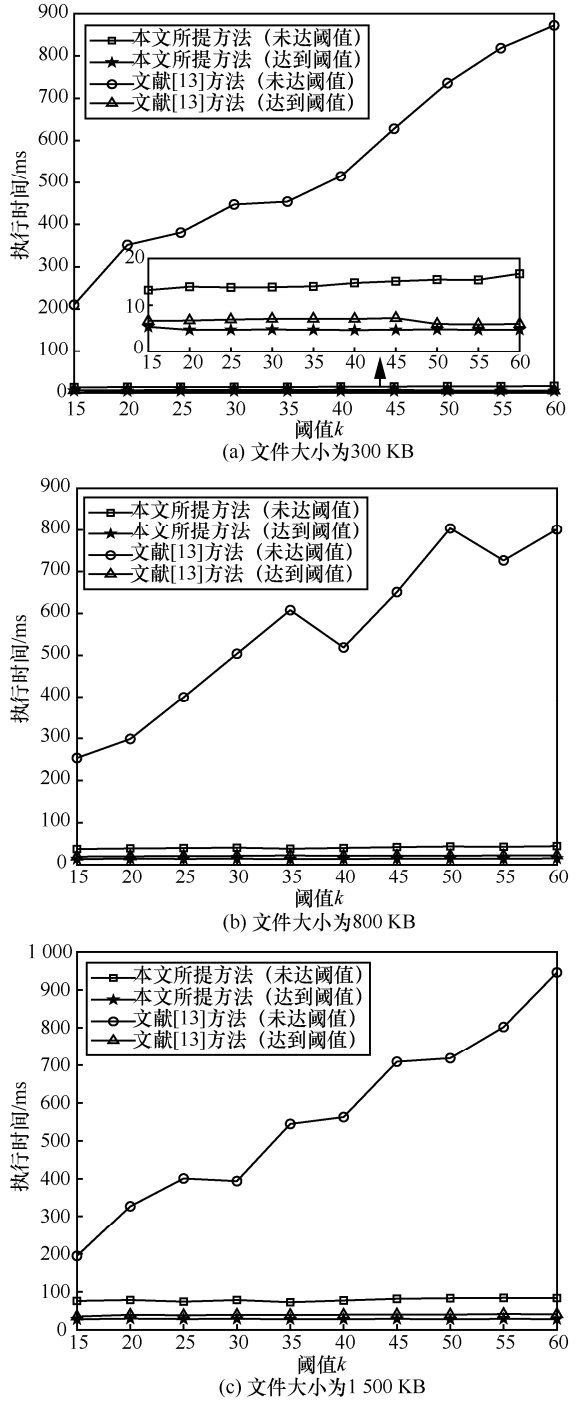
本节实验比较所提方法和文献[13]方法在不同阈值 k 情况下的用户端计算开销，采用算法平均执行时间来衡量计算开销。为了保证比较条件的一致性，将文件大小分别固定为 300 KB、800 KB 和 1 500 KB，选取的最低比特 m=4，比较阈值 k 取不同值时，所提方法和文献[13]方法的平均执行时间，结果如图 2 所示。

由图 2 可以看到，除文件大小为 300 KB，文献[13]方法（达到阈值）外，其他算法平均执行时间随阈值 k 增大基本保持上升趋势。这是因为 k 越大，所提方法用户端需要执行的加法运算次数、乘法运算次数和指数运算次数均随之增加，而文献[13]方法的加法、乘法、指数及幂指数运算次数均增加。由于取模的不确定性，所提方法未达阈值时与文献[13]方法相比优势不明显。综合来看，所提方法的平均执行时间显著低于文献[13]方法。

6.3　云端存储开销

本节验证所提方法的去重效果，结果用云端存储开销来衡量。以固定大小的文件 F 为例，当文件大小分别为 300 KB 和 1 500 KB 时，衡量所提方法

在阈值 k=20 和 k=40 的情况下云端存储开销随上传用户数量变化的情况，结果表示在对数坐标系中。为了排除其他因素的影响，只比较了引入所提的去重机制带来的存储开销变化，结果如图 3 所示。



(a) 文件大小为300 KB

(b) 文件大小为800 KB

(c) 文件大小为1 500 KB

图 2　不同阈值 k 下算法平均执行时间比较

由图 3 可以看到，当文件 F 的上传用户数量未达到阈值时，随着上传用户数量的增加，云端存储开销均显著增加。这是因为所提方法在云端文件副本数量未达到阈值 k 时，要求用户上传基于其 UID 生成的密文副本及 $N_{\mathrm{UID\_}F}$、验证信息 $(\mathrm{tag}_2, N_{\mathrm{VF}})$、数据查询信息 $(\mathrm{tag}_1, N_{\mathrm{QF}})$，同时云端需要保存用户的 UID 信息，这些信息随着上传用户数量的增加而增大。而一旦云端文件副本数量达到了阈值 k，云端存储开销迅速下降且基本保持恒定，因为云端只需保存重加密密文、标签信息和用户 ID 信息。

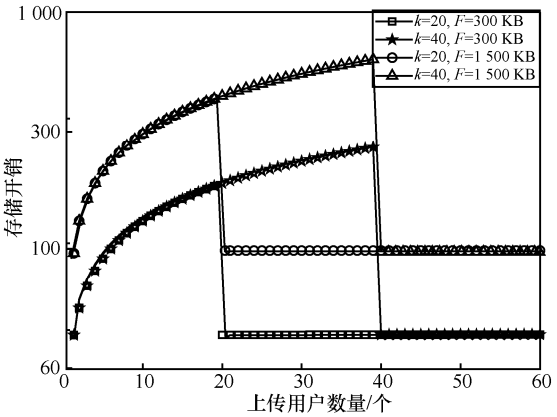


图 3　云端存储开销随上传用户数量变化的关系

7　结束语

本文提出了一种基于阈值重加密的抗边信道攻击云数据安全去重方法。在所提方法下，仅当同一文件的云端密文副本数达到或超过阈值时，才由云端执行去重，从而避免了攻击者通过边信道攻击窃取指定用户数据存在性隐私的安全风险。特别地，本文提出了一种轻量级的阈值重加密机制，将用户端的密文分割转变为密钥分割，并且把二次加密转移到云端执行，大大减少了用户端的计算开销。所提机制允许用户从加密密文和重加密密文中均可解密出明文，避免了用私钥加密同一文件的开销。同时，所提方法支持云服务提供商和用户端双向的数据完整性验证，直接确保密文副本和用户端明文数据的对应性。实验结果表明，所提方法在用户端计算开销、云端存储开销等方面均取得了很好的性能。

参考文献：

[1] GAI K K, QIU M K. Blend arithmetic operations on tensor-based fully homomorphic encryption over real numbers[J]. IEEE Transactions on Industrial Informatics, 2017, 14(8): 3590-3598.

[2] LIU J, ASOKAN N, PINKAS B. Secure deduplication of encrypted data without additional independent servers[C]//Proceedings of the 22nd ACM Conference on Computer and Communications Security.

New York: ACM Press, 2015: 874-855.

[3] YAN Z, DING W X, YU X X, et al. Deduplication on encrypted big data in cloud[J]. IEEE Transactions on Big Data, 2016, 2(2): 138-150.

[4] BELLARE M, KEELVEEDHI S, RISTENPART T. DepLess: server-aided encryption for deduplicated storage[C]//Proceedings of the 22nd USENIX Security Symposium. Berkeley: USENIX Association, 2013: 179-194.

[5] KWON H, HAHN C, KOO D Y, et al. Scalable and reliable key management for secure deduplication in cloud storage[C]//Proceedings of IEEE the 10th International Conference on Cloud Computing. Piscataway: IEEE Press, 2017: 391-398.

[6] DUAN Y. Distributed key generation for encrypted deduplication: achieving the strongest privacy[C]//Proceedings of the 21st ACM Conference on Computer and Communications Security Workshop. New York: ACM Press, 2014: 57-68.

[7] YU C M. Poster: efficient cross-user chunk-level client-side data deduplication with symmetrically encrypted two-party interactions[C]//Proceedings of the 23rd ACM Conference on Computer and Communications Security. New York: ACM Press, 2016: 1763-1765.

[8] ZUO P F, HUA Y, WANG C, et al. Mitigating traffic-based side channel attacks in bandwidth-efficient cloud storage[C]// Proceedings of the 32nd IEEE International Parallel & Distributed Processing Symposium. Piscataway: IEEE Press, 2018: 1153-1162.

[9] POORANIAN Z, CHEN K C, YU C M, et al. RARE: defeating side channels based on data-deduplication in cloud storage[C]//Proceedings of the 37th IEEE International Conference on Computer Communications Workshops. Piscataway: IEEE Press, 2018: 444-449.

[10] YU C M, GOCHHAYAT S P, CONTI M, et al. Privacy aware data deduplication for side channel in cloud storage[J]. IEEE Transactions on Cloud Computing, 2018, doi: 10.1109/TCC.2018.2794542.

[11] HARNIK D, PINKAS B, SHULMAN-PELEG A. Side channels in cloud services: deduplication in cloud storage[J]. IEEE Security & Privacy, 2010, 8(6): 40-47.

[12] STANEK J, KENCL L. Enhanced secure thresholded data deduplication scheme for cloud storage[J]. IEEE Transactions on Dependable and Secure Computing, 2018, 15(4): 694-707.

[13] ZHANG Y, MAO Y L, XU M Z, et al. Towards thwarting template side-channel attacks in secure cloud deduplications[J]. IEEE Transactions on Dependable and Secure Computing, 2019, doi: 10.1109/TDSC. 2019.2911502.

[14] STANEK J, SORNIOTTI A, ANDROULAKI E, et al. A secure data deduplication scheme for cloud storage[C]//Proceedings of the 18th International Conference on Financial Cryptography and Data Security. S.n.: s.l., 2014: 99-118.

[15] ARMKNECHT F, BOYD C, DAVIES G T, et al. Side channels in deduplication: Trade-offs between leakage and efficiency[C]//Proceedings of the 12nd ACM Asia Conference on Computer and Communications Security. New York: ACM Press, 2017: 266-274.

[16] BELLARE M, KEELVEEDHI S, RISTENPART T. DepLESS: server-aided encryption for deduplicated storage[C]//Proceedings of the 22nd USENIX Security Symposium. Berkeley: USENIX Association, 2013: 179-194.

[17] LIU X F, SUN W H, LOU W J, et al. One-tag checker: message-locked integrity auditing on encrypted cloud deduplication storage[C]//Proceedings of the 36th IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2017: 1-9.

[18] TANG X, ZHOU L N, HUANG Y F, et al. Efficient cross-user deduplication of encrypted data through re-encryption[C]//Proceedings of the 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications. Piscataway: IEEE Press, 2018: 897-904.

[19] DANG H, CHANG E C. Privacy-preserving data deduplication on trusted processors[C]//Proceedings of IEEE the 10th International Conference on Cloud Computing. Piscataway: IEEE Press, 2017: 66-73.

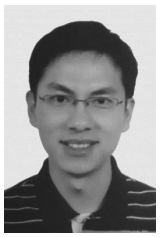
[20] ZHANG K, LIANG X H, LU R X, et al. Sybil attacks and their defenses in the Internet of Things[J]. IEEE Internet of Things Journal, 2014, 1(5): 372-383.

[21] DOUCEUR J, ADYA A, BOLOSKY W, et al. Reclaiming space from duplicate files in a server-less distributed file system[C]//Proceedings of the 22nd International Conference on Distributed Computing Systems. Piscataway: IEEE Press, 2002: 617-624.

[22] SHAMIR A. How to share a secret[J]. Communications of the ACM, 1979, 22(11): 612-613.

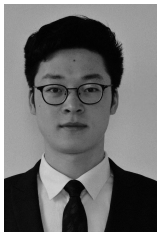
[23] ATENIESE G, HOHENBERGER. Proxy re-signatures: new definitions, algorithms, and applications[C]//Proceedings of the 22nd ACM Conference on Computer and Communications Security. New York: ACM Press, 2015: 310-319.

[作者简介]



唐鑫（1987– ），男，江苏南京人，博士，国际关系学院讲师，主要研究方向为数字内容安全、信息隐藏、云数据安全。

周琳娜（1972– ），女，湖南衡阳人，博士，北京邮电大学教授，主要研究方向为信息内容安全、行为分析、数字取证、信息隐藏。



单伟杰（1996– ），男，河南周口人，国际关系学院硕士生，主要研究方向为信息隐藏、云数据安全。

刘丹（1995– ），女，河北石家庄人，国际关系学院硕士生，主要研究方向为信息隐藏、云数据安全。