加密去重场景下基于 AONT 和 NTRU 的密钥更新方案

贾春福[1,2]，哈冠雄[1,2]，武少强[1,2]，陈杭[1,2]，李瑞琪[1,2]
（1. 南开大学网络空间安全学院，天津 300350；2. 天津市网络与数据安全技术重点实验室，天津 300350）

摘 要：密钥更新是对抗密钥泄露的有效方法。现有加密去重系统大多基于消息锁加密实现，拥有相同数据的多个用户共享同一加密密钥，某一用户更新密钥时其他数据所有者需同步该更新，这将引起较大的计算和通信开销。针对这一问题，提出了一种基于 AONT 和 NTRU 的密钥更新方案，设计了一个 AONT 的变体以解决多用户密钥更新时的同步问题，引入了一种基于 NTRU 的代理重加密方案以降低密钥更新过程中的系统通信开销和客户端计算开销。效率分析与实验结果表明，所提方案与现有方案相比具有更高的加解密效率，显著降低了密钥更新过程中的时间开销。

关键词：云存储；加密去重；密钥更新；AONT；NTRU
中图分类号：TP309.2
文献标识码：A
DOI: 10.11959/j.issn.1000−436x.2021187

AONT-and-NTRU-based rekeying scheme for encrypted deduplication

JIA Chunfu[1,2], HA Guanxiong[1,2], WU Shaoqiang[1,2], CHEN Hang[1,2], LI Ruiqi[1,2]
1. College of Cyber Science, Nankai University, Tianjin 300350, China
2. Tianjin Key Laboratory of Network and Data Security Technology, Tianjin 300350, China

Abstract: Rekeying is a good way to protect against key exposure. Most of the existing encrypted deduplication systems are implemented based on message-locked-encryption, in which multiple users with the identical data share the same encryption key. When a user updates keys, that update must be followed by all other data owners, which will incur large computational and communicational overheads. To solve this problem, an AONT-and-NTRU-based rekeying scheme was proposed, a variant of AONT was designed to solve the synchronization problem of multi-user rekeying, and a proxy re-encryption algorithm based on NTRU was introduced to reduce the communicational overhead for the system and computational overhead for clients during rekeying. The efficiency analysis and experimental results show that the proposed scheme has better encryption and decryption efficiency than existing schemes and the time cost of rekeying is significantly reduced.

Keywords: cloud storage, encrypted deduplication, rekeying, AONT, NTRU

1 引言

云计算的飞速发展让个人和企业都越来越倾向于将数据外包至云服务器存储以节省本地的数据存储和管理开销[1-2]。面对海量的外包数据，如何实现高效的数据存储成为云服务提供商面临的

收稿日期：2021−05−31；**修回日期**：2021−08−31
通信作者：贾春福，cfjia@nankai.edu.cn
基金项目：国家重点研发计划基金资助项目（No.2018YFA0704703）；国家自然科学基金资助项目（No.61972215, No.61972073, No.62172238）；天津市自然科学基金资助项目（No.20JCZDJC00640）
Foundation Items: The National Key Research and Development Program of China (No.2018YFA0704703), The National Natural Science Foundation of China (No.61972215, No.61972073, No.62172238), The Natural Science Foundation of Tianjin (No.20JCZDJC00640）

关键问题。数据去重[3-4]是提高存储利用率的有效方法，当收到多份相同的数据副本时，云服务器可通过数据去重仅存储其中的非重复内容以节省存储开销。对用户而言，出于对外包数据的机密性和隐私考虑，更希望能够将数据在本地加密后上传至云服务器，防止自己的数据信息被泄露。然而，在传统的加密算法中，每个用户使用自己的密钥加密数据，不同用户间的相同数据将被加密为不同的密文，云服务器无法进行去重。

为此，文献[5]提出了消息锁加密（MLE, message-locked encryption），其中加密密钥基于数据内容生成，拥有相同数据的用户可生成相同的加密密钥，此特性可用于实现跨用户的加密去重。但 MLE 存在拥有相同数据的多个用户共享同一加密密钥的特点，使多用户间的密钥更新缺乏独立性。系统中某一数据所有者进行密钥更新后，其他数据所有者均需同步该更新[6]，这为外包数据的密钥更新带来了困难。密钥更新[7-9]是密钥泄露后有效维持外包数据机密性，以及为外包数据提供有效访问控制[10-11]的重要方法。缺少密钥更新的云存储系统将在密钥泄露后的系统健壮性，以及在外包数据的访问控制变更等方面存在严重不足。

收敛加密[12]（CE, convergent encryption）是 MLE 中一个最常用的实例，其以数据的哈希值作为加密密钥。下面以收敛加密为例说明基于 MLE 的加密去重系统难以进行密钥更新的原因。假设云服务器存储的某一外包数据为 F，加密密钥为数据本身的哈希值 $H(F)$，其中 $H(\cdot)$ 表示哈希函数。当 $H(F)$ 泄露后，F 的数据所有者需要更新加密密钥，防止同时截获密钥 $H(F)$ 和外包密文 C_F 的敌手获得数据信息。密钥更新的过程如图 1 所示，数据所有者首先需要将原密文从云服务器端下载解密得到 F；然后选择一个新的哈希函数 $H'(\cdot)$ 作为新的 MLE 密钥生成函数，生成新的加密密钥 $H'(F)$，使用新密钥重新加密数据上传至云服务器；最后还需要将 $H'(\cdot)$ 广播至其他全部数据所有者。其他数据所有者同样需要完成下载外包数据，重新计算新 MLE 密钥的过程以保证密钥更新后系统仍可进行跨用户数据去重。因此，基于 MLE 的加密去重系统的密钥更新过程非常烦琐，并且具有较大的计算和通信开销。

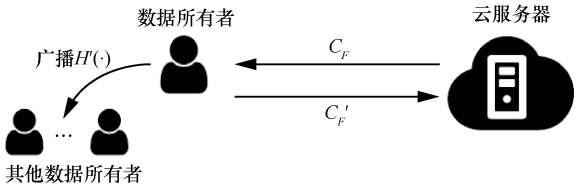


图 1 CE 中的密钥更新

针对上述问题，本文提出了一种加密去重场景下基于 AONT 和 NTRU 的密钥更新方案 ANRDup，设计了一个全有或全无转换[13-14]（AONT, all-or-nothing transform）的变体以解决多用户数据去重时密钥更新的同步问题，引入了一种基于 NTRU 的代理重加密方案[15]以降低密钥更新过程中的系统通信开销和客户端计算开销。ANRDup 利用 AONT 将用户数据拆分为用于数据去重的修剪包和用于密钥更新的存根。基于 AONT 的全有或全无特性，密钥更新时客户端仅需重加密外包数据的存根部分，而用于数据去重的修剪包可维持不变，这一特性实现了不同用户密钥更新时的独立性，避免了多用户密钥更新时的同步问题。ANRDup 中还引入了一种基于 NTRU 的代理重加密方案，密钥更新时客户端仅需计算并上传一个重加密密钥至云服务器，大大提升了密钥更新的执行效率。本文的主要贡献总结如下。

1) 设计了一个加密去重场景下的 AONT 的变体，安全高效地将整体外包数据的密钥更新转换到了体量较小的存根的密钥更新上，解决了基于 MLE 的加密去重系统中多用户密钥更新的同步问题。

2) 在密钥更新的过程中引入了一种基于 NTRU 的代理重加密方案，将计算量较大的密文转换过程外包至计算能力更强的云服务器完成，可显著降低密钥更新过程中的系统通信开销和客户端计算开销。

3) 分析了 ANRDup 的正确性、安全性和效率，并对 ANRDup 的原型进行了性能评估。评估结果表明 ANRDup 与现有方案[16]相比具有更高的加解密效率，并且显著降低了密钥更新过程中的时间开销。

2 预备知识

2.1 MLE

MLE 是加密去重系统中一个常用的密码原语，其加密密钥基于数据内容生成。这一特性使相

同数据对应的 MLE 密钥相同，保证了相同数据在加密后可得到相同的密文，便于云服务器检测密文的重复性。MLE 密文需要对应一个数据标签，用于检测数据重复。MLE 一般由以下几种算法组成。

1) KeyGen(M)：MLE 密钥生成算法输入用户数据 M，输出其对应的 MLE 密钥 K。

2) Enc(K, M)：MLE 加密算法输入 MLE 密钥 K 和用户数据 M，输出 M 加密后的密文 C。

3) Dec(K, C)：MLE 解密算法输入 MLE 密钥 K 和密文 C，输出 C 解密后的明文 M。

4) TagGen(C)：标签生成算法输入密文 C，输出其对应的数据标签 T。

MLE 中的 KeyGen() 和 TagGen() 一般使用哈希函数将用户数据和密文映射到体量较小的密钥和标签中，加解密算法则一般使用传统的对称加密算法（如 AES）。由于 MLE 中的加密密钥是基于数据内容生成而非随机生成，其无法为用户数据提供语义安全。只有在数据不可预测，即敌手无法遍历明文空间的情况下才可保证数据的机密性[17]。

2.2 AONT

AONT 是一个随机加密模式，其特性是只有得到全部密文后才可恢复出数据信息，缺少任一部分的密文都无法正确解密。假设数据 M 被拆分为 $\{M_1, M_2, \cdots, M_n\}$，随机密钥为 K。AONT 转换后的前 n 个包为 $C_i = M_i \oplus \mathrm{Enc}_{\mathrm{SE}}(K, i+1)$，其中，$i = 1,2,3,\cdots,n$，$\mathrm{Enc}_{\mathrm{SE}}(\cdot)$ 表示对称加密算法，$\oplus$ 表示异或，第 $n+1$ 个包为 $C_{n+1} = K \oplus H(C_1 \| C_2 \| \cdots \| C_n)$。当恢复数据时，解密方必须得到全部 $n+1$ 个包才能恢复出密钥 $K = C_{n+1} \oplus H(C_1 \| C_2 \| \cdots \| C_n)$ 并解密密文。AONT 的性质类似于一个 $(n+1, n+1)$ 的秘密共享[18]。除非得到全部 $n+1$ 个包，否则任何数据信息都无法恢复。

2.3 NTRU

NTRU[19]是一种基于格的密码算法。在众多公钥加密算法中，其加密效率较为出众且可用于构造代理重加密方案。如 NTRUReEncrypt[15]就是一种高效的基于 NTRU 的代理重加密方案。

这里介绍一些 NTRU 中的相关定义和概念。令 $\Phi(x)=x^n+1$，其中，n 为 2 的幂次，$\Phi(x)$ 为分圆多项式。q 为一个素数，满足 $q = 1 \bmod 2n$。R 为环 $\mathbb{Z}[x]/\Phi(x)$，$R_q = R/qR = \mathbb{Z}_q[x]/\Phi(x)$。NTRU 的加密方案定义在环 R 和 R_q 上，将 R_q 中的可逆元素集合定义为 $R_q^{\times}$，消息空间 M 设定为环 $R_p = R/pR$，其中，$p \in R_q^{\times}$。NTRU 一般由以下几种算法组成。

1) KeyGen()。密钥生成算法从高斯分布中取样 f' 和 g，令 $f = pf' + 1$，若 $(f \bmod q) \notin R_q^{\times}$ 或 $(g \bmod q) \notin R_q^{\times}$，则重新取样。计算 $h = pgf^{-1}$，输出私钥 $\mathrm{sk} = f$ 和公钥 $\mathrm{pk} = h$。

2) Enc(pk, m)。加密算法输入公钥 pk 和消息 $m \in M$，从高斯分布中取样噪声多项式 s 和 e，输出密文 $c = hs + pe + m \in R_q$。

3) Dec(sk,c)。解密算法输入私钥 sk 和密文 c，输出消息 $m = ((\mathrm{sk}\, c) \bmod p) \in M$。

NTRUReEncrypt 在 NTRU 的基础上加入了重加密密钥生成算法 ReKeyGen() 和重加密算法 ReEnc()，设计了一种基于 NTRU 的代理重加密方案，其由以下几种算法组成。

1) KeyGen()。对于用户 A 来说，密钥生成算法从高斯分布中取样 f_A' 和 g_A，令 $f_A = pf_A' + 1$，若 $(f_A \bmod q) \notin R_q^{\times}$ 或 $(g_A \bmod q) \notin R_q^{\times}$，则重新取样。计算 $h_A = pg_A f_A^{-1}$，输出私钥 $\mathrm{sk}_A = f_A$ 和公钥 $\mathrm{pk}_A = h_A$。

2) ReKeyGen(sk_A, sk_B)。重加密密钥生成算法输入用户 A 的私钥 sk_A 和用户 B 的私钥 sk_B，输出用户 A 和 B 之间的重加密密钥 $\mathrm{rk}_{A\to B} = \mathrm{sk}_A \mathrm{sk}_B^{-1} = f_A f_B^{-1} \bmod q$。

3) Enc(pk_A, m)。加密算法输入公钥 pk_A 和消息 $m \in M$，从高斯分布中取样噪声多项式 s 和 e，输出密文 $c_A = h_A s + pe + m \in R_q$。

4) ReEnc($\mathrm{rk}_{A\to B}$, c_A)。重加密算法输入重加密密钥 $\mathrm{rk}_{A\to B}$ 和用户 A 的密文 c_A，从高斯分布中取样噪声多项式 e'，输出新密文 $c_B = c_A \mathrm{rk}_{A\to B} + pe' \in R_q$。

5) Dec(sk_A, c)。解密算法输入私钥 sk_A 和密文 c_A，输出消息 $m = ((\mathrm{sk}_A c_A) \bmod p) \in M$。

3 相关工作

3.1 加密去重

针对传统加密算法与数据去重间的矛盾，文献[5]提出了 MLE，其使用基于数据内容生成的加密密钥构造确定性加密以实现跨用户的加密去重。但由于 MLE 的加密过程缺乏随机性，当用户数据可预测时，易受到敌手的离线字典攻击[17]。为此，DupLESS[17]引入了一个协助客户端生成加密密钥的密钥服务器以构造服务器辅助 MLE。DupLESS

中的密钥生成需要客户端与密钥服务器交互，敌手无法进行离线字典攻击，并且，系统可通过在密钥服务器端限制与客户端的交互速率防止敌手进行在线字典攻击。但中心化的密钥服务器易成为系统中的单点故障和效率瓶颈。因此，一些现有工作[20-22]设计了无第三方服务器的加密去重方案。但其中的客户端交互过多，并且要求大量客户端在线，难以适用于真实场景。

此外，加密去重系统中的安全问题还包括侧信道攻击[23-24]、所有权欺骗攻击[6,25]、频率分析攻击[26]、故障容错问题[27]和完整性审计[28-29]等。本文关注的主要问题是加密去重系统中的密钥更新问题。

3.2 密钥更新

在加密去重系统存在的众多安全问题当中，密钥更新问题近年来得到了越来越多的关注。现有的密钥更新方案可大致分为对加密密钥的密文进行密钥更新和对外包数据本身进行密钥更新 2 类。

3.2.1 密钥密文的密钥更新

分层加密是实现加密去重系统中密钥更新的重要方法。客户端首先使用 MLE 密钥加密用户数据，然后使用一个用户密钥对 MLE 密钥进行加密得到 MLE 密钥的密文。密钥更新时仅对密钥的密文进行更新。EDedup[30]利用密钥分层的思想实现密钥更新，系统中的加密密钥可分为段级别和文件级别，其中，段级别密钥用于加密用户数据，而文件级别密钥用于加密段级别密钥。系统通过重加密文件级别密钥来实现密钥更新。文献[31]提出了将用户角色与加密密钥相关联的思路，实现了分层结构下的云数据去重。该方案通过撤销角色树的节点实现密钥更新，主要思路是对密钥分层结构的调整。文献[32]利用策略加密保护 MLE 密钥的安全性。在执行密钥更新时，该方案结合公钥加密和身份认证的思想实现密钥密文的重加密。文献[10]使用 ElGamal 算法加密对称密钥，密钥更新时通过重新拆分 ElGamal 私钥，并将其外包至分布式密钥服务器以防止敌手通过已泄露的密钥破坏数据机密性。

基于分层加密设计的密钥更新方案的特点是系统仅对密钥的密文进行密钥更新。当外层密钥泄露时，方案可保证数据机密性；但当直接加密用户数据的内层密钥泄露后，外包数据的机密性将受到破坏。

3.2.2 数据密文的密钥更新

由于对密钥的密文进行密钥更新这一方法具有局限性，一些研究工作开始关注对外包数据本身进行密钥更新。一些可更新块级别消息锁加密（UMLE, updatable block-level message-locked encryption）的相关研究工作[33-34]试图在文件的数据块动态改变时高效地更新 MLE 密钥。UMLE 属于对外包数据本身进行密钥更新的方法，但其研究重点主要在于数据内容的变化对于 MLE 密钥的影响，而非系统应如何对抗 MLE 密钥泄露。因此，在 UMLE 中，当敌手得到已泄露的 MLE 密钥和外包数据时，系统无法为用户数据提供机密性保证。

REED[16]提出了支持密钥更新的 2 种加密去重方案：基本加密方案和增强加密方案，其主要思想是通过使用 AONT 将用户数据拆分为修剪包和存根，然后通过对存根进行重加密实现外包数据的密钥更新。该方案实现了外包数据本身的密钥更新，可有效防止得到已泄露密钥的敌手破坏数据机密性。然而，其提出的 2 种方案均存在一定的缺陷：在基本加密方案中，一旦 MLE 密钥泄露，数据机密性将不能得到保证；增强加密方案为提高安全性需要对用户数据进行 2 层加密，这带来了额外的计算开销。此外，在 REED 的密钥更新中，存根数据需要由客户端从云服务器端下载、重加密并上传。当存根体量较大或密钥更新频繁时，仍会引起不小的计算和通信开销。

4 系统模型

4.1 应用场景

ANRDup 的应用场景为某一群组的用户（如公司内的全体员工或高校内的各个院系）共同外包数据至云服务器，如图 2 所示。用户不完全信任云服务器，将自己的数据加密后外包，防止云服务器窥探其数据隐私。由于使用云存储服务的用户均属于同一群组，其外包数据中很可能会存在大量的重复内容，云服务器可使用去重技术节省存储开销。一旦外包数据的加密密钥泄露，数据所有者可执行密钥更新操作防止同时得到已泄露的加密密钥和外包密文的敌手获取用户数据信息。

4.2 系统架构

ANRDup 中包括 3 种实体：客户端、密钥服务器和云存储服务器（简称为云服务器）。

客户端：用户通过客户端外包数据至云服务器。客户端将用户数据拆分为数据块，对其进行加密，将密文块上传至云服务器。客户端可在密钥泄露后更新其外包数据的加密密钥。

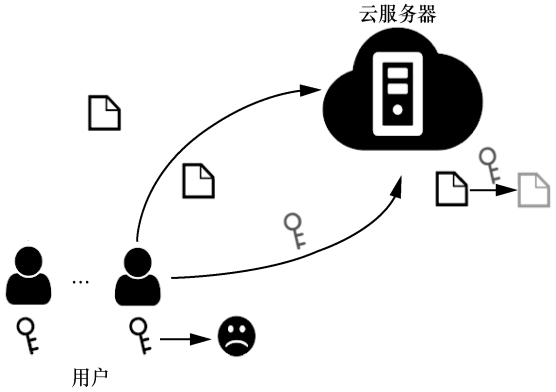


图 2　应用场景

密钥服务器：为了弥补 MLE 难以为可预测数据提供机密性的缺陷，ANRDup 中部署了密钥服务器以实现服务器辅助 MLE。客户端生成 MLE 密钥时需要与密钥服务器交互，MLE 密钥的生成同时基于数据内容与密钥服务器管理的系统主密钥。

云服务器：云服务器可为多个用户提供数据存储和管理服务，并可使用去重技术降低其存储开销。

4.3　敌手模型

本文主要考虑 2 类敌手：内部敌手和外部敌手。内部敌手为诚实但好奇的云服务器，可通过外包数据和已泄露的用户私钥试图获取数据信息。外部敌手为系统外的恶意用户，可得到已泄露的 MLE 密钥和用户外包数据。

本文的敌手模型基于如下的安全假设。首先，假设客户端与云服务器间的通信基于 SSL/TLS 协议，二者在身份认证后进行通信。其次，由于系统可在密钥服务器端设定与客户端的交互速率限制，本文不考虑敌手的在线字典攻击，并且，方案中客户端与密钥服务器间的交互基于不经意的伪随机函数（OPRF, oblivious pseudo-random function），由于 OPRF 的安全性已被证明[17]，本文假设密钥服务器无法在密钥生成阶段获取用户的数据信息。

基于上述的敌手模型，ANRDup 主要有以下 2 个安全目标。

1) 为用户的外包数据提供机密性，即上述的内部或外部敌手均无法获取用户的数据信息。

2) 为用户的外包数据提供完整性，即客户端从云服务器下载数据后可检测外包数据是否被篡改。

5　方案设计

5.1　设计思想

通过对现有加密去重系统的研究与分析，总结了其进行密钥更新的 2 个难点。

1) 在基于 MLE 的加密去重系统中，拥有相同数据的多个用户共享同一加密密钥，使当某个数据所有者对其外包数据进行密钥更新时，其他数据所有者必须同步该更新，否则将出现相同数据加密后无法去重的情况。

2) 当需要更新密钥的文件体量较大时，大量数据需要在客户端和云服务器间传输，这将引起较大的通信开销。此外，重加密大体量文件也给计算能力本就相对薄弱的客户端带来了较大的计算开销。

针对第一个难点，本文构造一个适用于加密去重场景下的 AONT 的变体来转换用户数据。首先，将原有 AONT 中的随机密钥替换为基于数据内容生成的确定性密钥，构造收敛 AONT[27]（CAONT, convergent all-or-nothing transform）。拥有相同数据的不同用户可计算得到相同的确定性密钥，因此跨用户间的相同数据经 AONT 转换后可得到相同的包数据以便实现加密去重。ANRDup 中的数据加密过程的方案设计如图 3 所示，客户端首先将用户文件拆分为数据块 $\{M_1,M_2,\cdots,M_n\}$，然后将这些数据块通过 AONT 转换为修剪包 $\{\mathrm{tp}_1,\mathrm{tp}_2,\cdots,\mathrm{tp}_n\}$ 和存根 $\{\mathrm{st}_1,\mathrm{st}_2,\cdots,\mathrm{st}_n\}$，其中，修剪包与用户数据大小相同，用于云服务器进行去重处理；存根则需要进一步由 NTRU 随机加密后得到密文存根，用于客户端进行密钥更新。由于不同用户的 NTRU 密钥不同，云服务器无法对跨用户的密文存根进行去重，但由于其体量较小，不会引起过多的存储开销。

假设 t 个用户 $\{U_1,U_2,\cdots,U_t\}$ 经加密去重后共享同一密文 C。当某个用户 U_i 进行密钥更新时，其仅需重加密外包数据的存根部分即可，而用于数据去重的修剪包部分维持不变，这维持了跨用户数据去重的特性，其他用户不需要同步 U_i 的密钥更新，解决了上述的第一个难点。由于 AONT 的“全有或全无”特性，解密方缺少密文的存根部分便无法恢复数据信息。因此，系统可通过重加密存根实现整体外包数据的密钥更新。

针对第二个难点，避免密钥更新时体量较大的外包数据在客户端与云服务器间传输以及在客户端完成计算量较大的加解密操作的方法是将密文

转换的过程直接外包至云服务器完成。此外，密文转换的过程中需要保证云服务器无法获取用户的数据信息。代理重加密正是实现这一需求的有效方法。由于 ANRDup 中使用 NTRU 加密存根，密钥更新的过程中可引入基于 NTRU 的代理重加密方案 NTRUReEncrypt[15]。客户端使用新旧密钥生成一个重加密密钥，将其上传至云服务器，云服务器完成数据的重加密过程，具体的方案设计如图 3 所示。云服务器使用该重加密密钥和原存根密文通过重加密过程生成新的存根密文。因此，在整个密钥更新的过程中，系统的通信开销仅为一个重加密密钥，客户端的计算开销仅为生成该重加密密钥，这解决了上述的第二个难点。

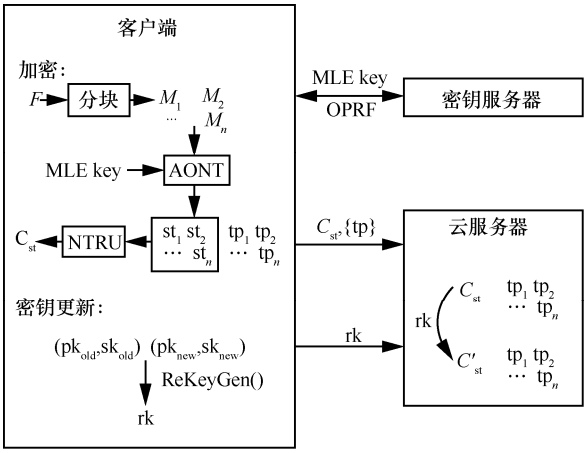


图 3 方案设计

5.2 方案细节

本节主要介绍 ANRDup 中的服务器辅助 MLE 密钥生成，数据加解密和密钥更新的详细过程。

5.2.1 服务器辅助 MLE 的密钥生成

在 ANRDup 中，客户端加密数据前需要与密钥服务器交互生成 MLE 密钥，这里使用 RSA-OPRF[17] 生成 MLE 加密密钥。具体过程如下。

1) 密钥服务器生成 RSA 公私钥对 $\{\mathrm{pk}=(e,N),\mathrm{sk}=d\}$，将公钥 pk 分发至所有客户端，安全存储私钥 sk。

2) 客户端计算用户数据的哈希值 h，从群 $\mathbb{Z}_N^*$ 中随机选取用于盲化的随机值 $r\leftarrow_R\mathbb{Z}_N^*$，计算 $x=hr^e \bmod N$ 发送至密钥服务器。

3) 密钥服务器利用私钥对客户端发送的数据签名，得到 $y=x^d \bmod N$。

4) 客户端进行去盲化过程，计算 $z=yr^{-1}=h^d \bmod N$，验证 h 是否与 $z^e \bmod N$ 相等。若相等，说明该 MLE 密钥 z 有效；否则该密钥无效，需重新生成。

5.2.2 数据加密

ANRDup 中 AONT 的确定性密钥同时基于数据块内容和服务器辅助 MLE 密钥生成，此设计是为了防止出现类似于 REED 的基本加密方案中 MLE 密钥泄露后数据机密性被破坏的情况。此外，为了在密钥更新中引入基于 NTRU 的代理重加密方案，客户端使用 NTRU 加密存根。图 4 描述了 ANRDup 的数据加密流程。

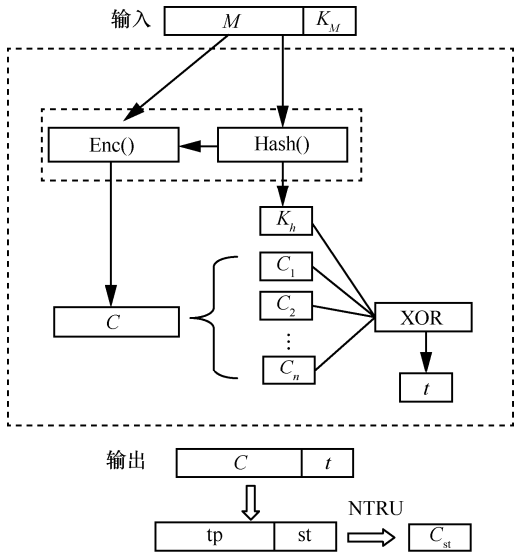


图 4 数据加密流程

下面以加密单个数据块 M 为例详细介绍数据加密过程。假设 M 对应的服务器辅助 MLE 密钥为 K_M，方案的加密过程如下。

1) 拼接 M 和 K_M 得到 $(M\|K_M)$，计算加密密钥 $K_h=H(M\|K_M)$。

2) 使用 K_h 加密 $(M\|K_M)$ 得到密文 $C=\mathrm{Enc}_{\mathrm{SE}}(K_h,(M\|K_M))$，本文的对称加密使用了 AES 加密算法。

3) 将密文 C 拆分为 $\{C_1,C_2,\cdots,C_n\}$，其中每份密文均与密钥 K_h 大小相同。异或 n 份密文和 K_h 得到 $t=C_1\oplus C_2\oplus\cdots\oplus C_n\oplus K_h$。

4) 拼接 C 和 t 得到 $(C\|t)$，从中截取部分数据（如 64 byte）作为存根 st，剩余部分作为修剪包 tp。

5) 生成 NTRU 密钥对 (pk,sk)，使用公钥 pk 加密存根 st 得到 $C_{\mathrm{st}}=\mathrm{Enc}_{\mathrm{NTRU}}(\mathrm{pk},\mathrm{st})$，其中，$\mathrm{Enc}_{\mathrm{NTRU}}(\cdot)$ 为 NTRU 加密算法。

不同用户上传相同数据块时计算得到的确定

性密钥相同，因此得到的修剪包也相同，云服务器可对跨用户上传的修剪包进行数据去重。此外，本文设计的加密方案是一个 AONT 的变体，满足计算安全性。除非敌手得到了全部的 AONT 包数据或者可暴力猜测出加密密钥，否则其无法得到数据块的任何信息。6.2 节对该问题给出了更详尽的安全分析。

5.2.3 数据解密

客户端从云服务器下载数据块 M 的全部密文（即存根密文 C_{st} 和修剪包 tp）后，可通过以下步骤完成解密操作以恢复 M。

1) 使用 NTRU 私钥 sk 解密 C_{st} 得到存根 $\mathrm{st}=\mathrm{Dec}_{\mathrm{NTRU}}(\mathrm{sk},C_{\mathrm{st}})$，其中 $\mathrm{Dec}_{\mathrm{NTRU}}(\cdot)$ 为 NTRU 解密算法。

2) 拼接 tp 和 st 得到完整的 AONT 包数据 $(\mathrm{tp}\,\|\,\mathrm{st})$，从中截取与 K_h 大小相同的字节得到 t，将剩余内容作为密文 C。

3) 将密文 C 拆分为 $\{C_1,C_2,\cdots,C_n\}$，与 t 异或得到 $K_h=C_1\oplus C_2\oplus\cdots\oplus C_n\oplus t$。

4) 利用 K_h 解密密文 C，得到 $(M\,\|\,K_M)=\mathrm{Dec}_{\mathrm{SE}}(K_h,C)$，从中截取得到数据块 M，其中，$\mathrm{Dec}_{\mathrm{SE}}(\cdot)$ 为对称解密算法。

5) 计算 $(M\,\|\,K_M)$ 的哈希值 $h=H(M\,\|\,K_M)$，比较 h 和密钥 K_h 是否相等，若相等则数据完整性未被损坏。否则，数据完整性已被损坏。

5.2.4 密钥更新

当 NTRU 密钥对泄露时，数据所有者需要对外包数据进行密钥更新，具体过程如下。

1) 假设泄露的密钥对 $(\mathrm{pk}_{\mathrm{old}},\mathrm{sk}_{\mathrm{old}})$，客户端生成新的 NTRU 密钥对 $(\mathrm{pk}_{\mathrm{new}},\mathrm{sk}_{\mathrm{new}})$，输入新旧私钥至重加密密钥生成算法 $\mathrm{RekeyGen}(\cdot)$，计算得到重加密密钥 $\mathrm{rk}_{\mathrm{old}\to\mathrm{new}}=\mathrm{RekeyGen}(\mathrm{sk}_{\mathrm{old}},\mathrm{sk}_{\mathrm{new}})$，发送至云服务器。

2) 云服务器输入重加密密钥 $\mathrm{rk}_{\mathrm{old}\to\mathrm{new}}$ 和原密文 $C_{\mathrm{st}}=\mathrm{Enc}_{\mathrm{NTRU}}(\mathrm{pk}_{\mathrm{old}},\mathrm{st})$ 至重加密算法 $\mathrm{ReEnc}(\cdot)$，输出新密文 $C'_{\mathrm{st}}=\mathrm{ReEnc}(\mathrm{rk}_{\mathrm{old}\to\mathrm{new}},C_{\mathrm{st}})$，其中，$C'_{\mathrm{st}}$ 仅可由 $\mathrm{sk}_{\mathrm{new}}$ 解密，而 $\mathrm{sk}_{\mathrm{old}}$ 失效。

5.3 方案流程

本节详细介绍 ANRDup 中客户端的数据上传、数据恢复和密钥更新的详细操作流程。

5.3.1 数据上传

假设用户的外包文件为 F，客户端首先将其拆分为大量数据块 $\{M\}$，计算这些数据块的哈希值，基于 OPRF[35]与密钥服务器交互得到 $\{M\}$ 对应的 MLE 密钥 $\{K_M\}$。$\{M\}$ 经 AONT 转换后得到存根 $\{\mathrm{st}\}$ 和修剪包 $\{\mathrm{tp}\}$，客户端将文件 F 对应的所有存根 $\{\mathrm{st}\}$ 写入存根文件 F_{st}。然后客户端生成 NTRU 密钥对 $(\mathrm{pk},\mathrm{sk})$，对 F_{st} 进行 NTRU 加密得到密文存根 $C_{\mathrm{st}}=\mathrm{Enc}_{\mathrm{NTRU}}(\mathrm{pk},F_{\mathrm{st}})$。由于 NTRU 中的加密单元为多项式，客户端需要在加密前将存根文件编码为多项式系数，7.2.1 节对编码方式进行了详细介绍。最后，客户端将修剪包 $\{\mathrm{tp}\}$ 和密文存根 C_{st} 上传至云服务器，本地存储 NTRU 密钥对 $(\mathrm{pk},\mathrm{sk})$ 用于解密数据；云服务器存储客户端上传的密文存根和修剪包，并可对跨用户上传的修剪包进行去重。

5.3.2 数据恢复

客户端从云服务器下载得到修剪包 $\{\mathrm{tp}\}$ 和密文存根 C_{st}，使用 NTRU 私钥 sk 解密 C_{st} 得到 $F_{\mathrm{st}}=\mathrm{Dec}_{\mathrm{NTRU}}(\mathrm{sk},C_{\mathrm{st}})$，使用 $\{\mathrm{tp}\}$ 和 F_{st} 恢复文件 F。若在恢复文件时检测到外包数据已被篡改，则解密操作终止。

5.3.3 密钥更新

客户端生成新的 NTRU 密钥对 $(\mathrm{pk}_{\mathrm{new}},\mathrm{sk}_{\mathrm{new}})$，基于原密钥对中的私钥 $\mathrm{sk}_{\mathrm{old}}$ 和新私钥 $\mathrm{sk}_{\mathrm{new}}$ 生成重加密密钥 $\mathrm{rk}_{\mathrm{old}\to\mathrm{new}}$，将 $\mathrm{rk}_{\mathrm{old}\to\mathrm{new}}$ 上传至云服务器。云服务器利用原密文存根 C_{st} 和重加密密钥 $\mathrm{rk}_{\mathrm{old}\to\mathrm{new}}$ 生成新的密文存根 C'_{st}。

6 安全性分析

本节首先分析 ANRDup 的正确性，即数据加密或密钥更新后，客户端可正确恢复用户数据。然后，分析了 ANRDup 中设计的 AONT 的安全性，并基于 4.3 节中的敌手模型分析了整体方案的安全性。

6.1 正确性分析

ANRDup 中所使用的对称加密的正确性已被证明，这里基于文献[15]对 NTRU 的加解密和重加密过程进行正确性分析，假设客户端加密的存根数据为 m。

NTRU 加密后得到的密文为 $c=hs+pe+m\in R_q$，解密时计算 $c\mathrm{sk}=cf=(hs+pe+m)f=(pgf^{-1}s+pe+m)f=pgs+pef+mf \bmod p=mf \bmod p$。由于 $f=1 \bmod p$，$c\mathrm{sk}=mf=m \bmod p\in M$，因此，ANRDup 中 NTRU 的加密部分满足解密正确性。

假设原密文为 $c=hs+pe+m\in R_q$，新私钥为 $\mathrm{sk}'=f'$。NTRU 重加密后得到的密文为 $c'=c\mathrm{rk}+$

$pe' = pgf'^{-1}s + peff'^{-1} + mff'^{-1} + pe' \in R_q$。解密时可使用新私钥 sk′ 计算得到 $c'\mathrm{sk}' = pgs + pef + mf + pe'f' = m \bmod p \in M$。因此，ANRDup 中的重加密过程满足解密正确性。

6.2 AONT 的安全性分析

ANRDup 是基于 AONT 的全有或全无特性而设计的，全有或全无特性表示只有当得到 AONT 转换后的所有包数据（即修剪包和存根）时才可恢复数据信息，缺少任何一部分包数据均无法完成解密。这里以加密单个数据块 M 为例对 ANRDup 中所设计的 AONT 的全有或全无特性进行分析，假设 M 在 AONT 转换后可得到 $n+1$ 份密文 $C=\{C_1, C_2, \cdots, C_n, t\}$，加密密钥 K_h 的大小为 l_k。

定理 1 若 AES 具有选择明文攻击下的不可区分性，则方案中的 AONT 具有全有或全无特性。

证明 M 的密文块共 $n+1$ 份，假设敌手可得到其中的 n 份。分 2 种情况讨论，第一种即敌手得到了前 n 份数据 $\{C_1, C_2, \cdots, C_n\}$，第二种即敌手得到了 t 和 $\{C_1, C_2, \cdots, C_n\}$ 中的 $n-1$ 份。

1) 若敌手得到了 CN=$\{C_1, C_2, \cdots, C_n\}$，由于 CN 为 AES 密文，若 AES 具有选择明文攻击下的不可区分性，敌手无法区分 CN 与等长的随机比特串。因此，敌手依据 CN 恢复数据信息的概率等于其成功猜测出 AES 密钥 K_h 的概率，为 $1/2^{l^k}$，实现了计算安全性。若密钥长度为 128 位（即 $l_k=128$），敌手恢复数据信息的概率为 $1/2^{128}$，此概率是可忽略的。

2) 若敌手得到了 t 和 $\{C_1, C_2, \cdots, C_n\}$ 中的 $n-1$ 份，由于 ANRDup 中的 K_h 是经 $\{C_1, C_2, \cdots, C_n\}$ 异或后得到 t 的，这相当于对 K_h 进行了一次一密，为其提供了信息论安全。缺少 $\{C_1, C_2, \cdots, C_n\}$ 中任何一份的敌手恢复 K_h 的概率为 $1/2^{l^k}$，同样实现了计算安全性。

6.3 ANRDup 的安全性分析

本节首先针对敌手模型中的内部敌手对 ANRDup 进行了形式化的安全证明，将 ANRDup 的安全性规约到 AES 和 NTRUReEncrypt[15]等已被证明安全的密码学原语上。证明了若存在内部敌手可攻破 ANRDup，则该敌手可成功攻破 AES 和 NTRUReEncrypt。此外，本节证明了可得到 MLE 密钥的外部敌手同样无法恢复数据信息，并对方案中外包数据的完整性进行了分析。

定理 2 若方案中使用的 AES 和 NTRUReEncrypt 是密码学安全的，则内部敌手获取用户数据信息的概率是可忽略的。

证明 这里通过定义多个不可区分的安全游戏来证明定理 2。Game_0 模拟了真实场景下的 ANRDup 方案，敌手 $\mathcal{A}$ 模拟了诚实但好奇的内部敌手的行为。若 $\mathcal{A}$ 赢得 Game_0 的概率优势是可忽略的，则可认为方案中内部敌手获取数据信息的概率是可忽略的。

1) Game_0

初始化阶段。挑战者 $\mathcal{C}$ 生成 AES 密钥 k 和 NTRU 密钥对 $(\mathrm{pk}_0, \mathrm{sk}_0)$，将公钥 pk_0 发送至敌手 $\mathcal{A}$，安全存储 sk_0 和 k。假设初始化阶段为 e_0，挑战阶段为 e_C，猜测阶段为 e_G。在每个时段 e_i 开始前，$\mathcal{C}$ 生成新的密钥对 $(\mathrm{pk}_i, \mathrm{sk}_i)$，将此前时段 $e_{i\text{-}1}$ 的私钥 $sk_{i\text{-}1}$ 发送至 $\mathcal{A}$（挑战阶段的私钥 sk_C 不会发送给 $\mathcal{A}$）。

询问阶段 1。在 $e_0 \leqslant e_i < e_C$ 中，$\mathcal{A}$ 向 $\mathcal{C}$ 发起询问。$\mathcal{A}$ 输出消息 m，$\mathcal{C}$ 返回 m 在 ANRDup 中对应的密文 $C_m = (\mathrm{tp}, C_{\mathrm{st}})$。$\mathcal{C}$ 将 $\mathcal{A}$ 在这一阶段中输出的所有消息放入集合 Q_m。

挑战阶段。在 e_C 中，$\mathcal{A}$ 输出 2 个等长的挑战明文 m_0 和 m_1（$\{m_0, m_1\} \notin Q_m$），$\mathcal{C}$ 随机选择 $b \in_R \{0,1\}$，返回 m_b 对应的挑战密文 C_b。

询问阶段 2。在 $e_C < e_i < e_G$ 中，$\mathcal{A}$ 输出消息 m（$m \notin \{m_0, m_1\}$），$\mathcal{C}$ 返回 m 对应的密文 C_m。$\mathcal{A}$ 输出挑战密文 C_b，$\mathcal{C}$ 返回 C_b 经密钥更新后重加密为当前时段的密文 C_b'。若 $\mathcal{A}$ 在 e_i 向 $\mathcal{C}$ 询问 C_b，则 e_i 的私钥 sk_i 在游戏中将不会发送至 $\mathcal{A}$。

猜测阶段。在 e_G 中，$\mathcal{A}$ 输出 b'。如果 $b=b'$，则 $\mathcal{A}$ 赢得 Game_0。将 $\mathcal{A}$ 在 Game_0 中具有的概率优势定义为 $\mathrm{Adv}(\mathcal{A}) = |\Pr[b=b'] - \frac{1}{2}|$。

2) Game_1

与 Game_0 流程相同，区别在于 $\mathcal{C}$ 返回密文时使用一个与存根密文等长的随机比特串 $\mathrm{str}_{\mathrm{cst}}$，即 $\mathcal{C}$ 返回 $\mathcal{A}$ 询问的消息密文或重加密后的密文为 $C_m = (\mathrm{tp}, \mathrm{str}_{\mathrm{cst}})$。

3) Game_2

与 Game_1 流程相同，区别在于 $\mathcal{C}$ 返回密文时使用一个与修剪包等长的随机比特串 $\mathrm{str}_{\mathrm{tp}}$，即 $\mathcal{C}$ 返回 $\mathcal{A}$ 询问的消息密文或重加密后的密文为 $C_m = (\mathrm{str}_{\mathrm{tp}}, \mathrm{str}_{\mathrm{cst}})$。

将 $\mathcal{A}$ 在 Game_i 中输出 $b=b'$ 的事件定义为 S_i，$\mathcal{A}$ 在 Game_i 中可取得的概率优势定义为 $|\Pr[S_i]-\frac{1}{2}|$，其中 $i\in\{0,1,2\}$。

引理 1　$\Pr[S_2]=1/2$。

在 Game_2 中，$\mathcal{A}$ 得到的密文为 $C_m=(\text{str}_{\text{tp}},\text{str}_{\text{cst}})$，存根与修剪包均为与 m 无关的随机比特串，$\mathcal{A}$ 只能随机输出 b'。因此，$\Pr[S_2]=1/2$。

引理 2　若敌手在没有 AES 密钥的前提下区分 AES 密文与等长的随机比特串的概率为 ξ_A，则 $|\Pr[S_1]-\Pr[S_2]|\leqslant\xi_A$。

在 Game_1 中，密文为 $C_m=(\text{tp},\text{str}_{\text{cst}})$。$\mathcal{A}$ 区分 Game_1 与 Game_2 的概率将不超过其区分 AES 密文 tp 与等长的随机比特串 str_{tp} 的概率 ξ_A。

引理 3　若敌手在没有私钥的前提下区分 NTRUReEncrypt 中的密文与等长的随机比特串的概率为 ξ_N，则 $|\Pr[S_0]-\Pr[S_1]|\leqslant\xi_N$。

在 Game_0 中，$\mathcal{A}$ 可得到密文 $C_m=(\text{tp},C_{\text{st}})$ 以及 sk_C 外的私钥集合 $\{\text{sk}_0,\cdots,\text{sk}_{C-1},\text{sk}_C,\cdots,\text{sk}_{G-1}\}$。在没有 sk_C 的前提下，$\mathcal{A}$ 区分 Game_0 与 Game_1 的概率将不超过其区分密文 C_{st} 与随机比特串 str_{cst} 的概率 ξ_N。

由以上 3 条引理可知，$\mathcal{A}$ 赢得 Game_0 的概率优势如式(1)所示。由于 ξ_A 和 ξ_N 均为可忽略的值，$\mathcal{A}$ 赢得 Game_0 的概率优势是可忽略的。

$$\begin{aligned}\left|\Pr\left[S_0-\frac{1}{2}\right]\right|=&|\Pr[S_0]-\Pr[S_1]+\Pr[S_1]-\\&\Pr[S_2]+\Pr[S_2]-\frac{1}{2}\Big|\leqslant\big|\Pr[S_0]-\Pr[S_1]\big|+\\&\Big|\Pr[S_1]-\Pr[S_2]|+|\Pr[S_0]-\frac{1}{2}\Big|\leqslant\xi_A+\xi_N\end{aligned}\tag{1}$$

定理 3　得到外包数据和 MLE 密钥的外部敌手恢复用户数据信息的概率是可忽略的。

证明　外部敌手可同时得到 MLE 密钥和外包数据，ANRDup 可实现不可预测的数据块的机密性保证。方案中的加密密钥为 $K_h=H(M\,\|\,K_M)$，鉴于哈希函数中的雪崩效应（输入发生任何微小的改变，都会导致输出的不可区分性改变），除非外部敌手可得到完整的 $(M\,\|\,K_M)$，否则其无法计算得到 K_h。因此，外部敌手恢复数据信息的概率等于其暴力猜测密钥 K_h 的概率，即 $1/2^t$，此概率是可忽略的。

定理 4　客户端无法检测数据完整性被损坏的概率是可忽略的。

证明　客户端下载解密数据后可得到 $(M\,\|\,K_M)$，其可通过比较 $H(M\,\|\,K_M)$ 和 K_h 是否相等来判断数据完整性是否被损坏。由于哈希函数 $H(\cdot)$ 的抗碰撞性，数据被篡改后仍满足 $H(M\,\|\,K_M)=K_h$ 的概率是可忽略的。

7　性能分析

本节分别从理论分析与性能评估 2 个方面对 ANRDup 进行对比分析。

7.1　理论分析

本节从密钥更新中的客户端、服务器计算开销和系统中的通信开销 3 个方面对方案的效率进行理论分析，并比较了现有方案与 ANRDup 在密钥更新方式上的不同。表 1 展示了不同方案在密钥更新中的时间开销，其中 l_F 和 l_S 分别表示整体数据和存根的长度，l_{RK} 和 l_K 分别表示重加密密钥和对称密钥的长度，l_{Sig} 和 l_{AK} 分别表示方案中使用的签名长度和对称密钥经公钥加密后的密文长度。SE 和 SD 分别表示对称加解密的计算开销，AE 和 AD 分别表示公钥加解密的计算开销，H 和 oprf 分别表示哈希函数和 OPRF 引起的计算开销，RKG 和 RE 表示重加密密钥生成和数据重加密的计算开销。

表 1　密钥更新中的开销对比

方案	客户端计算开销	服务器端计算开销	带宽开销
DupLESS	$(\text{SD}+\text{SE}+H)O(l_F)+\text{oprf}$	—	$2l_F+2l_{\text{GD}}$
CE	$(\text{SD}+\text{SE}+H)O(l_F)$	—	$2l_F$
RCE	$(\text{SD}+H)O(l_F)+(\text{SD}+\text{SE})O(l_K)$	—	l_F+2l_K
文献[30]	$\text{RKG}O(l_{\text{RK}})$	$\text{RE}O(l_{\text{AK}})$	$l_{\text{AK}}+l_{\text{RK}}$
文献[32]	$\text{SD}O(l_K)+\text{AD}O(l_{\text{AK}})+\text{AE}O(l_K)$	$\text{AD}O(l_{\text{AK}})$	$2l_{\text{AK}}+2l_{\text{Sig}}$
REED	（SD+SE）$O(l_S)$	—	$2l_S$
ANRDup	$\text{RKG}O(l_{\text{RK}})$	$\text{RE}O(l_S)$	l_{RK}

由表 1 可以看出，CE、RCE 和 DupLESS 3 个经典的加密去重方案中密钥更新的开销较大，计算和通信开销均与外包数据的长度线性相关。与近年来的几种密钥更新方案[16,30,32]相比，ANRDup 将部分计算开销外包至服务器端，具有更低的客户端计算开销和通信开销。表 2 展示了现有方案在密钥更新方式上的不同。虽然文献[30]和文献[32]具有较高的密钥更新效率，但其密钥更新方式属于密钥密文的更新，当直接加密外包数据的密钥泄露时，无法保证用户数据的机密性。因此，只有 REED 与 ANRDup 实现了高效的数据密文更新，并且由表 1 可以看出，与 REED 相比，ANRDup 具有更低的客户端计算开销和带宽开销。

表 2　　密钥更新方式的对比

方案	数据密文更新	密钥密文更新
DupLESS	√	—
CE	√	—
RCE	—	√
文献[30]	—	√
文献[32]	—	√
REED	√	—
ANRDup	√	—

7.2 系统实现与性能评估

7.2.1 系统实现

本文基于开源的 REED 代码实现了 ANRDup 的原型，其中 AES 和哈希函数等密码学原语均基于 OpenSSL 实现，NTRU 加密则基于 NTL 库实现。由于 NTRU 的加密单元为多项式，客户端在对存根进行 NTRU 加密前需要首先将其编码为多项式系数。编码方式如下：客户端从存根中读取一定数量的比特串，将其编码为 ASCⅡ码。由于每个 ASCⅡ码占 8 位，且 NTRU 中的明文空间大小与参数 p 相关，客户端将每 p/8 个字符转换为一个 p bit 的整数，然后将其编码为多项式系数。客户端完成编码后，使用 NTRU 加密得到存根密文。

此外，本文对开源代码 REED 中 AONT 的实现进行了优化。为减少 AONT 中的比特间操作，方案的原型实现中将待处理数据的每 64 bit 分为一组执行运算，这使程序可在 64 位的机器上更高效地运行。

7.2.2 性能评估

本节分别使用人工数据和真实数据集对 ANRDup 的性能进行了评估。实验所使用的机器配备有 3.40GHz Inter i7-6700 处理器，3.5 GB RAM，64 位 Ubuntu 16.04.1 LTS。所有评估结果均是 10 次以上程序运行结果的平均值。人工数据包括一些内容随机填充的文件，真实数据集来自 FSLhomes，是石溪大学文件系统和存储实验室所收集的包含 9 个用户的每日备份的一个数据集，很多加密去重系统[16,26,32]的评估均基于此数据集。本节将 7.2.1 节中程序优化前后的 ANRDup 原型分别命名为 ANRDup_Basic 和 ANRDup_OP，将 REED 中的基本加密和增强加密方案分别命名为 REED_Basic 和 REED_Enhanced。

1) 人工数据上的评估

这里首先使用人工数据评估 ANRDup 在数据加密上传、下载解密和密钥更新 3 个方面的性能开销，其中测试文件的大小从 1 到 1 000 MB，4 种方案的 AONT 加解密开销分别如图 5(a)和图 5(b)所示。不难看出 ANRDup_OP 具有最好的 AONT 加解密效率。与其他 3 种方案相比，ANRDup_OP 分别降低了 45.3%～66.6%的 AONT 加密开销和 48.7%～72.5%的 AONT 解密开销，其性能提升的主要原因在于方案在 AONT 设计上的改进，以及代码实现上的优化。REED_Enhanced 的 AONT 加解密效率最低，原因在于其对用户数据进行了 2 层加密。ANRDup_Basic 和 REED_Basic 的 AONT 加解密效率相近。

图 6(a)和图 6(b)显示了 ANRDup 和 REED 在加解密存根上的时间开销对比。由于在 ANRDup_OP 和 ANRDup_Basic 中存根的加解密过程相同，且均是基于 NTRU 加密算法实现的，因此统一简称为 ANRDup_NTRU。而 REED 中的 2 种加密方案均是基于 AES 加密算法实现的，因此统一简称为 REED_AES。由于 AES 的加解密速度明显优于 NTRU，在加解密存根的时间开销上 ANRDup 要高于 REED。然而，由于存根的体量较小，其加解密的开销不会对整体的数据加密上传或下载解密产生过大的影响，这一点在图 7 中得到了证明。

图 7 显示了 ANRDup 和 REED 在整体的数据加解密上的时间开销对比，图 7(a)和图 7(b)分别对应数据加密和解密的时间开销，其中包括了 AONT 过程和存根加解密过程中的时间开销。ANRDup_OP 在整体的数据加解密上与其他 3 个方案相比分别降低了 19.6%～49.4%和 37.1%～57.0%

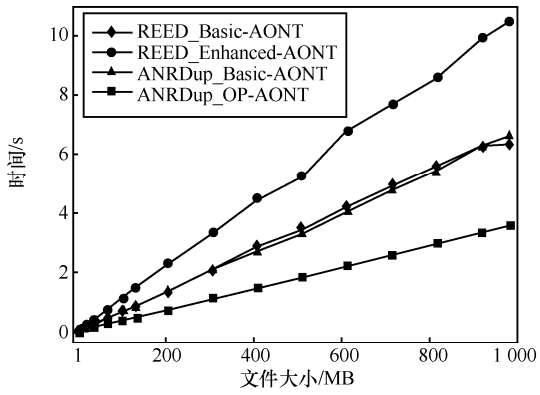


图 5　AONT 时间开销

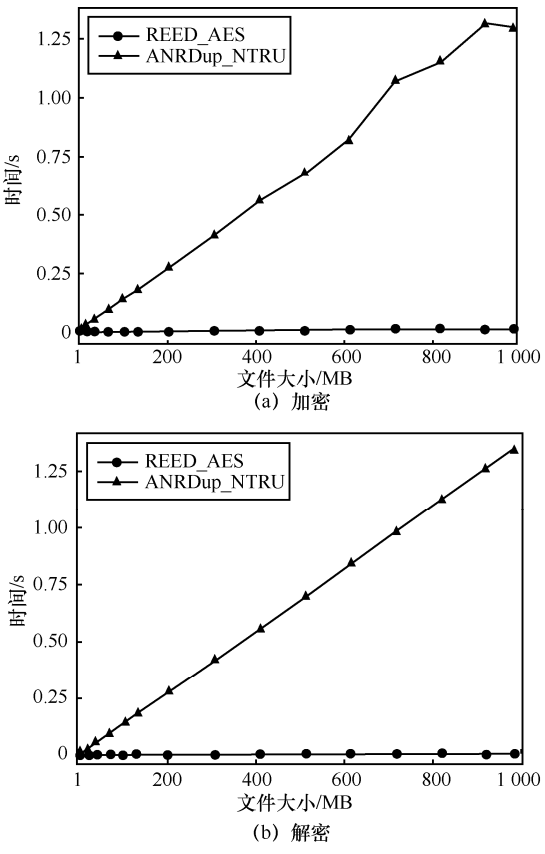


图 6　存根加解密的时间开销

的时间开销。ANRDup_Basic 在解密速度上与 REED_Basic 相似，加密速度略慢于 REED_Basic，但与安全性相同的 REED_Enhanced 相比，ANRDup_Basic 具有更好的加解密效率。整体的数据加解密开销的对比情况充分说明了 ANRDup 中 NTRU 的引入并未对数据加解密的效率产生过大的影响，主要原因在于由 NTRU 加解密的存根体量较小且方案在 AONT 的设计和实现上均实现了优化。

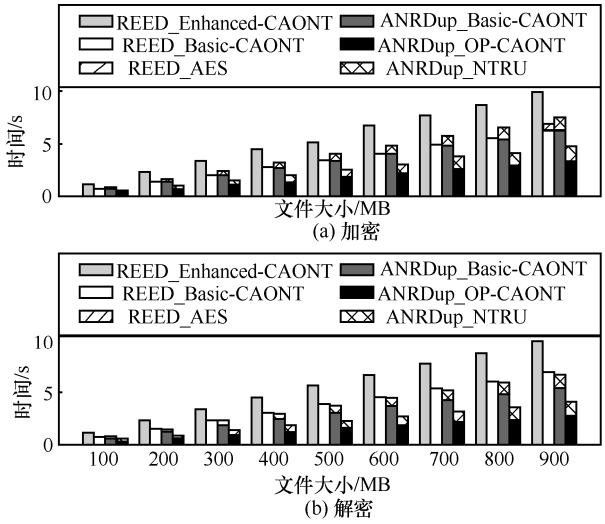
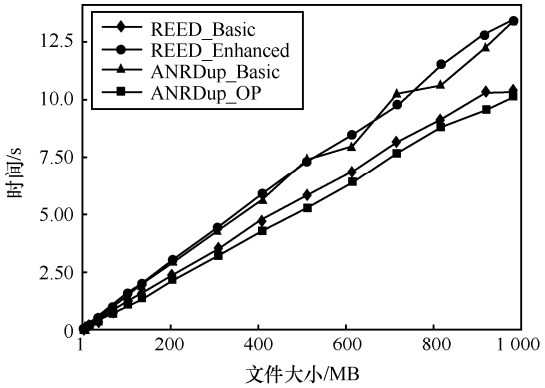


图 7　整体数据加解密的时间开销

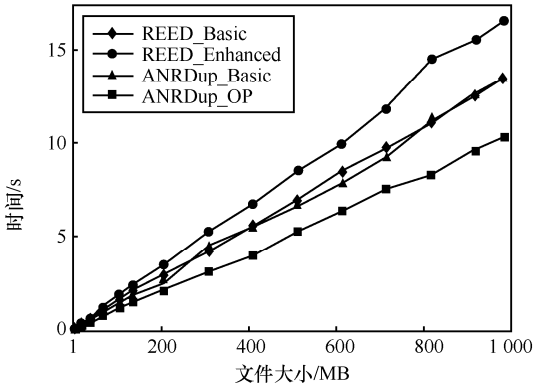
图 8(a)和图 8(b)比较了 ANRDup 与 REED 在数据上传下载过程中客户端的时间开销。与图 7 相比，图 8 中增加了文件分块和数据通信部分的开销，反映了系统真实运行时客户端的时间开销。ANRDup_OP 在数据上传和下载过程中与其他 3 种方案相比分别降低了 21.3%～22.1%和 23.3%～38.8%的时间开销。

ANRDup 与 REED 在性能上最大的差距体现在密钥更新的过程上，这同样也是本方案设计的主要目标。由于 2 个版本的 ANRDup 的密钥更新开销相同，统一称为 ANRDup。出于相同的原因，REED 中基本加密和增强加密统一称为 REED。2 种方案密钥更新过程中的性能差异主要集中在系统通信开销和客户端的计算开销上。图 9 比较了 2 种方案在密钥更新过程中的时间开销，REED 中密钥更新的时间开销随文件大小的增加而增长，而 ANRDup 中的时间开销是几乎恒定的。原因在于在 REED 的密钥更新中客户端需要下载密文存根、解密、重加密后上传至云服务器。文件体量越大，其存根的体量也会随之增加。因此，REED 中密钥更新的时间

开销与文件大小线性相关，而 ANRDup 中的数据重加密过程则外包至云服务器，无论需要密钥更新的文件大小如何，客户端的计算开销都仅为生成一个重加密密钥，系统的通信开销都仅为该重加密密钥的大小，二者均与文件大小无关。



(a) 上传



(b) 下载

图 8 上传下载的时间开销

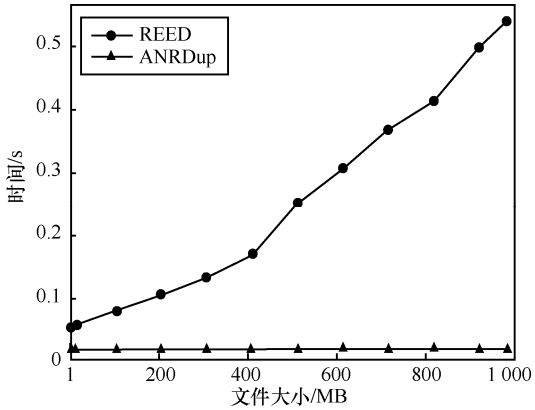
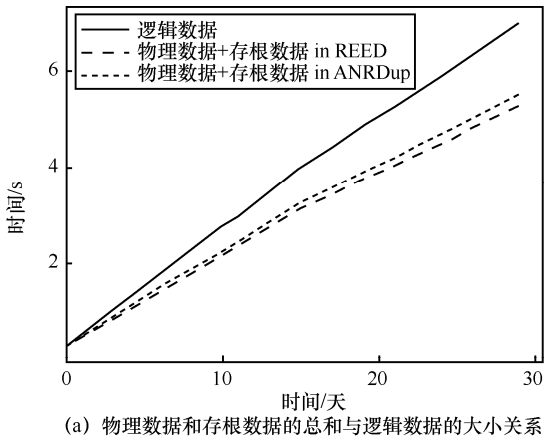


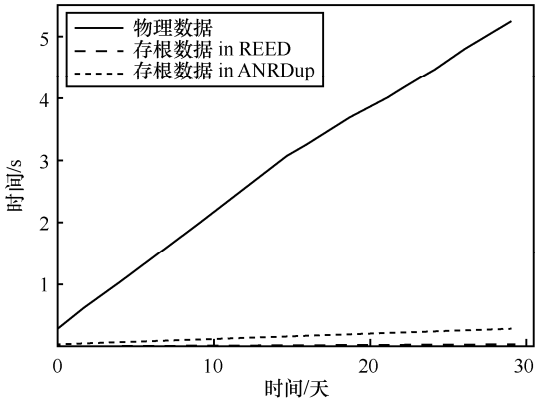
图 9 密钥更新的时间开销

2) 真实数据集上的评估

本小节使用真实数据集 FSLhomes 对 ANRDup 的存储效率和加解密性能进行了评估。图 10 中的逻辑数据（logical data）表示未经任何加密和去重的原始数据，物理数据（physical data）表示去重后存储在云服务器的修剪包，存根数据（stub data）表示云服务器存储的存根数据。尽管 ANRDup 并未对存根数据进行去重，且 NTRU 加密后的存根存在一定的密文扩张，但 ANRDup 仍具有较高的存储效率。图 10(a)比较了 ANRDup 和 REED 中物理数据和存根数据的总和与逻辑数据的大小关系，可以看出数据去重后 2 种方案的存储效率均得到了提高。存储连续 29 天的备份数据后，ANRDup 和 REED 的去重率分别为 75.6%和 78.8%，其中，去重率定义为物理数据和存根数据的总和除以逻辑数据的比率。由此可以看出，ANRDup 中 NTRU 所引起的密文扩张并未对方案整体的存储开销带来过大的影响。图 10(b)比较了 ANRDup 和 REED 中物理数据和存根数据分别所占的存储开销。与物理数据相比，存根数据所占的存储开销比例较低。ANRDup 中存根数据占总存储量（即物理数据和存根数据的总和）的 5%，而 REED 中该比例为 1%。二者间的差异主要来自 NTRU 加密算法引起的密文扩张。



(a) 物理数据和存根数据的总和与逻辑数据的大小关系



(b) 物理数据和存根数据的存储开销

图 10 存储开销

图 11 展示了 ANRDup_OP 和与其安全性相同的 REED_Enhanced 在真实数据集上的加解密性能。性能测试选择了 5 个用户连续 6 天的日常备份数据。这 6 天中 ANRDup_OP 的平均加解密速度分别为 73.83 Mbit/s 和 57.12 Mbit/s，REED_Enhanced 为 58.42 Mbit/s 和 43.13 Mbit/s。ANRDup_OP 在真实数据集上的加解密速度比 REED_Enhanced 分别提高了 20.9%和 24.5%。

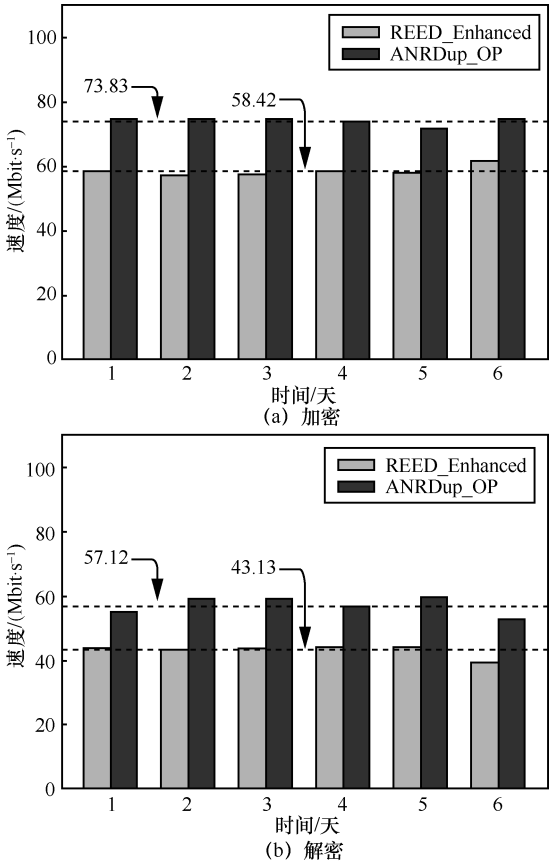


图 11　真实数据集的加解密开销

8　结束语

针对现有加密去重系统的密钥更新过程中存在的困难，提出了基于 AONT 和 NTRU 的密钥更新方案 ANRDup。方案基于 AONT 的全有或全无特性实现了不同用户密钥更新时的独立性，解决了基于 MLE 的加密去重系统中多用户密钥更新时存在的同步问题。同时，方案在密钥更新中引入了一种基于 NTRU 的代理重加密方案降低了系统的通信开销和客户端的计算开销。性能评估表明所提方案具有较高的加解密效率，并且显著降低了密钥更新过程中的时间开销。由于方案使用了第三方的密钥服务器，易产生单点故障和效率瓶颈的问题。后续工作可通过设计基于阈值的分布式密钥服务器以缓解该问题。

参考文献：

[1] 冯登国, 张敏, 张妍, 等. 云计算安全研究[J]. 软件学报, 2011, 22(1): 71-83.
FENG D G, ZHANG M, ZHANG Y, et al. Study on cloud computing security[J]. Journal of Software, 2011, 22(1): 71-83.

[2] 熊金波, 张媛媛, 李凤华, 等. 云环境中数据安全去重研究进展[J]. 通信学报, 2016, 37(11): 169-180.
XIONG J B, ZHANG Y Y, LI F H, et al. Research progress on secure data deduplication in cloud[J]. Journal on Communications, 2016, 37(11): 169-180.

[3] SHIN Y, KOO D, HUR J. A survey of secure data deduplication schemes for cloud storage systems[J]. ACM Computing Surveys, 2017, 49(4): 1-38.

[4] XIA W, JIANG H, FENG D, et al. A comprehensive study of the past, present, and future of data deduplication[J]. Proceedings of the IEEE, 2016, 104(9): 1681-1710.

[5] BELLARE M, KEELVEEDHI S, RISTENPART T. Message-locked encryption and secure deduplication[C]//Advances in Cryptology – EUROCRYPT 2013. Berlin: Springer, 2013: 296-312.

[6] XU J, CHANG E C, ZHOU J Y. Weak leakage-resilient client-side deduplication of encrypted data in cloud storage[C]//Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security. New York: ACM Press, 2013: 195-206.

[7] BOYD C, DAVIES G T, GJØSTEEN K, et al. Fast and secure updatable encryption[C]//Advances in Cryptology – CRYPTO 2020. Cham: Springer International Publishing, 2020: 464-493.

[8] LEHMANN A, TACKMANN B. Updatable encryption with post-compromise security[C]//Advances in Cryptology – EUROCRYPT 2018. Cham: Springer International Publishing, 2018: 685-716.

[9] JARECKI S, KRAWCZYK H, RESCH J. Updatable oblivious key management for storage systems[C]//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2019: 379-393.

[10] 贾春福, 哈冠雄, 李瑞琪. 密文去重系统中的数据访问控制策略[J]. 通信学报, 2020, 41(5): 72-83.
JIA C F, HA G X, LI R Q. Data access control policy of encrypted deduplication system[J]. Journal on Communications, 2020, 41(5): 72-83.

[11] LI J, CHEN X F, LI J W, et al. New access control systems based on outsourced attribute-based encryption1[J]. Journal of Computer Security, 2015, 23(6): 659-683.

[12] DOUCEUR J R, ADYA A, BOLOSKY 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.

[13] RESCH J K, PLANK J S. AONT-RS: blending security and performance in dispersed storage systems[C]//Proceedings of the 9th USENIX Conference on File and Storage Technologies. San Jose: USENIX Association, 2011: 191-202.

[14] RIVEST R L. All-or-nothing encryption and the package transform[C]//Proceedings of Fast Software Encryption. Berlin: Springer, 1997: 210-218.
[15] NUÑEZ D, AGUDO I, LOPEZ J. NTRUReEncrypt: an efficient proxy Re-encryption scheme based on NTRU[C]//Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security. New York: ACM Press, 2015: 179-189.
[16] QIN C, LI J W, LEE P P C. The design and implementation of a re-keying-aware encrypted deduplication storage system[EB]. arXiv: 1607.08388. 2016.
[17] BELLARE M, KEELVEEDHI S, RISTENPART T. DupLESS: server-aided encryption for deduplicated storage[J]. IACR Cryptology EPrint Archive, 2013, 2013: 429.
[18] BEIMEL A. Secret-sharing schemes: A survey[C]//Proceedings of International Conference on Coding and Cryptology. Berlin: Springer, 2011: 11-46.
[19] HOFFSTEIN J, PIPHER J, SILVERMAN J H. NTRU: A ring-based public key cryptosystem[C]//Proceedings of the International Algorithmic Number Theory Symposium. Berlin: Springer, 1998: 267-288.
[20] LIU J, ASOKAN N, PINKAS B. Secure deduplication of encrypted data without additional independent servers[C]//Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2015: 874-885.
[21] LIU J, DUAN L, LI Y, et al. Secure deduplication of encrypted data: refined model and new constructions[C]//Proceedings of Cryptographers' Track at the RSA Conference. Cham: Springer International Publishing, 2018: 374-393.
[22] YU C M. POSTER: efficient cross-user chunk-level client-side data deduplication with symmetrically encrypted two-party interactions[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2016: 1763-1765.
[23] 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.
[24] POORANIAN Z, CHEN K C, YU C M, et al. RARE: Defeating side channels based on data-deduplication in cloud storage[C]//Proceedings of IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). Piscataway: IEEE Press, 2018: 444-449.
[25] HALEVI S, HARNIK D, PINKAS B, et al. Proofs of ownership in remote storage systems[C]//Proceedings of the 18th ACM conference on Computer and communications security. New York: ACM Press, 2011: 491-500.
[26] LI J W, LEE P P C, TAN C F, et al. Information leakage in encrypted deduplication via frequency analysis[J]. ACM Transactions on Storage, 2020, 16(1): 1-30.
[27] LI M Q, QIN C, LI J W, et al. CDStore: toward reliable, secure, and cost-efficient cloud storage via convergent dispersal[C]//Proceedings of IEEE Internet Computing. Piscataway: IEEE Press, 2016: 45-53.
[28] LI J W, LI J, XIE D Q, et al. Secure auditing and deduplicating data in cloud[J]. IEEE Transactions on Computers, 2016, 65(8): 2386-2396.
[29] 郭晓勇, 付安民, 况博裕, 等. 基于收敛加密的云安全去重与完整性审计系统[J]. 通信学报, 2017, 38(S2): 156-163.
GUO X Y, FU A M, KUANG B Y, et al. Secure deduplication and integrity audit system based on convergent encryption for cloud storage[J]. Journal on Communications, 2017, 38(S2): 156-163.
[30] ZHOU Y K, FENG D, HUA Y, et al. A similarity-aware encrypted deduplication scheme with flexible access control in the cloud[J]. Future Generation Computer Systems, 2018, 84: 177-189.
[31] 熊金波, 张媛媛, 田有亮, 等. 基于角色对称加密的云数据安全去重[J]. 通信学报, 2018, 39(5): 59-73.
XIONG J B, ZHANG Y Y, TIAN Y L, et al. Cloud data secure deduplication scheme via role-based symmetric encryption[J]. Journal on Communications, 2018, 39(5): 59-73.
[32] XU R H, JOSHI J, KRISHNAMURTHY P. An integrated privacy preserving attribute-based access control framework supporting secure deduplication[J]. IEEE Transactions on Dependable and Secure Computing, 2021, 18(2): 706-721.
[33] ZHAO Y J, CHOW S S M. Updatable block-level message-locked encryption[J]. IEEE Transactions on Dependable and Secure Computing, 2021, 18(4): 1620-1631.
[34] LIU M Z, YANG C, JIANG Q, et al. Updatable block-level deduplication with dynamic ownership management on encrypted data[C]//Proceedings of 2018 IEEE International Conference on Communications (ICC). Piscataway: IEEE Press, 2018: 1-7.
[35] NAOR M, REINGOLD O. Number-theoretic constructions of efficient pseudo-random functions[C]//Proceedings of the 38th Annual Symposium on Foundations of Computer Science. Piscataway: IEEE Press, 1997: 458-467.

[作者简介]

贾春福（1967− ），男，河北文安人，博士，南开大学教授、博士生导师，主要研究方向为网络与信息安全、可信计算、恶意代码分析、密码技术应用等。

哈冠雄（1995− ），男，回族，天津人，南开大学博士生，主要研究方向为云数据安全、密码学应用。

武少强（1996− ），女，山西汾阳人，南开大学硕士生，主要研究方向为密码学应用、隐私保护。

陈杭（1998− ），女，天津人，南开大学硕士生，主要研究方向为密码学应用、加密去重。

李瑞琪（1993− ），男，黑龙江尚志人，南开大学博士生，主要研究方向为同态加密、格密码学等。