电子科技大学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

本科学位论文

BACHELOR THESIS

论文题目 密文重复数据删除机制的频率攻击方法

学科专业	信息安全
学　　号	2015040101018
作者姓名	任彦璟
指导老师	李经纬　副教授

分类号 _____ 密级 _____

UDC 注 1 _____

学　位　论　文

密文重复数据删除机制的频率攻击方法

（题名和副题名）

任彦璟

（作者姓名）

指导老师 李经纬　副教授

（姓名、职称、单位名称）

申请学位级别 本科 学科专业 信息安全

提交论文日期 _____ 论文答辩日期 _____

学位授予单位和日期 电子科技大学

答辩委员会主席 _____

评阅人 _____

注 1：注明《国际十进分类法 UDC》的类号。

摘　要

加密重复数据删除旨在解决大规模数据存储系统中的安全性和存储效率：它确保每个明文都由明文本身内容派生的对称密钥加密，从而为数据存储提供机密性保证，同时保留重复数据删除的存储节省效率。但是，加密重复数据删除的确定性特性也会泄漏明文的频率，从而使加密重复数据删除容易受到频率分析的影响。在本文中，我们重新审视了频率分析导致的加密重复数据删除的安全漏洞。我们认为加密重复数据删除可能更容易受到精心设计的频率分析攻击，因此攻击提供了高可信度，可确保每个推断的明文确实对应于目标密文的概率很高（即从统计角度来看，高精度）。为此，我们提出了两种新的频率分析攻击，它们可以适应实际重复数据删除工作负载的特性，从而提高频率分析的准确性。我们根据实际情况评估针对实际存储工作负载的建议攻击，并提供有关如何带来实际损失的观察。

关键词： 加密重复数据删除, 频率分析攻击, 聚类分析

ABSTRACT

Encrypted deduplication aims to address security and storage efficiency in large-scale data storage systems: it ensures that each plaintext is encrypted by a symmetric key that is derived from the content of the plaintext itself, so as to provide confidentiality guarantees for data storage while preserving the storage saving effectiveness of deduplication. However, the deterministic nature of encrypted deduplication also leaks the frequencies of plaintexts, thereby making encrypted deduplication vulnerable to frequency analysis. In this paper, we revisit the security vulnerability of encrypted deduplication due to frequency analysis. We argue that encrypted deduplication can be even more vulnerable to carefully crafted frequency analysis attacks, such that the attacks provide high confidence of ensuring that each inferred plaintext indeed corresponds to the target ciphertext with a high probability (i.e., high precision from a statistical perspective). To this end, we propose two new frequency analysis attacks that adapt the characteristics of practical deduplication workloads to increasing the severity of frequency analysis. We empirically evaluate our proposed attacks against real-world storage workloads, and provide observations on how they bring actual damages.

Keywords: Encrypted deduplication, Frequency analysis attacks, Cluster analysis

目 录

第一章 绪 论

1.1 研究工作的背景与意义

1.1.1 研究背景

随着信息技术产业的高速发展，数字信息量呈爆炸式增长。Gartner 研究表明[1]，仅 2015 年的移动数据流量就较 2014 年增长 59%；并且，这一增长率将持至 2018 年末，移动数据流量水平达 1.73 亿 TB。数据的快速增长导致企业面临的存储和管理成本越来越高[2]。另一方面，在存储系统所保存的数据中，高达 60%的数据都是冗余的，随着时间的推移，这些冗余数据的比例将进一步上升[3]。近年来，存储系统中数据高冗余的特点得到越来越多研究人员的关注，利用该特点来节省存储容量是一个热门研究课题。

数据重删技术（data deduplication）是指通过识别数据流中的冗余，只传输或存储唯一数据（unique data），而使用指向已存储数据的指针替换重复副本，以达到节省带宽或存储空间的目的[4]。由于能够有效地降低存储开销，数据重删技术非常适合为管理日益增长的海量数据节省成本。在工业界，EMC Data Domain[5] 和 Avamar[6]、Veritas 的 NetBackup Appliances[7] 以及 Commvault 的开放数据平台[8]都是比较知名的数据重删应用产品；此外，各大云存储厂商（例如 Dropbox、Google Drive、Bitcasa、Moza 等）也纷纷将数据重删技术应用于各自的云服务产品中，以提升经济效益[9]。

如图1-1所示，在支持数据重删的存储系统（统称为数据重删系统）中，重删后的任何数据块都被一个或多个文件引用，而文件则以指向这些数据块的指针的集合形式存储。这种文件共用数据块的存储模式强调了数据块的敏感性，因为一个数据块的泄漏可能扩散影响到共用这个数据块的所有文件。如何保护重删后的数据的隐私，成为信息安全领域的一个研究热点。

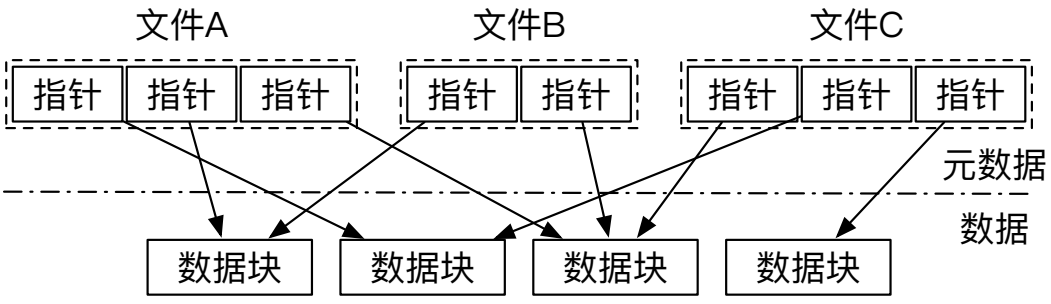


图 1-1 数据重删系统的存储模式

为了保护数据隐私，加密重复数据删除（encrypted deduplication）增加了一层作用于逻辑数据块的加密操作。如图1-2所示，该加密层基于数据内容来产生加密密钥[10]（例如将数据块的散列值作为密钥[11]），从而将相同的明文数据块加密为相同的密文数据块。系统计算每个密文数据块的散列值 (称为指纹，fingerprint)，查询指纹索引（fingerprint index）确定该数据块是否已经存储，最后保存仅具有唯一指纹的密文数据块。需要指出的是，部分加密重复数据删除方案[10]采用随机加密算法，但基于明文数据块产生指纹，因此仍然可以通过检查指纹来识别重复数据。

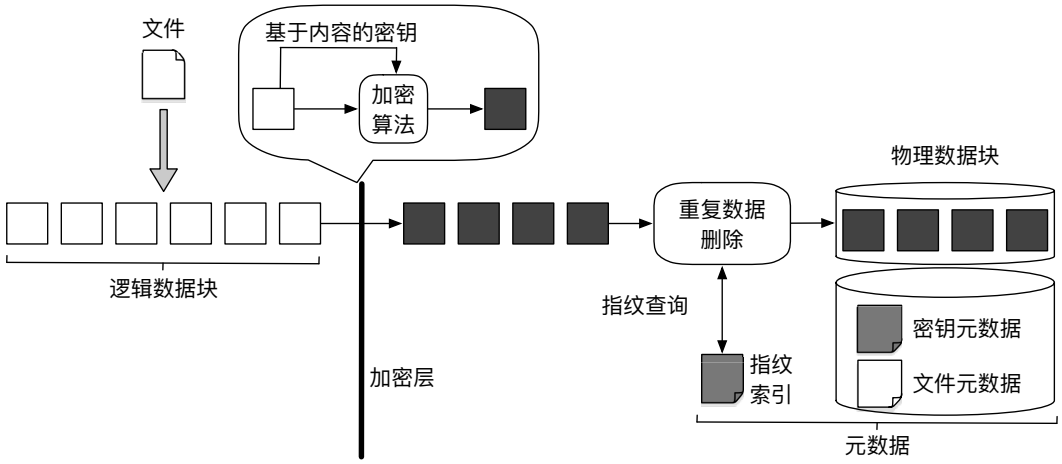


图 1-2 加密重复数据删除系统逻辑视图

除了指纹索引以外，加密重复数据删除系统须存储额外的元数据（metadata），包括:

1. 文件元数据记录了文件内逻辑数据块与相应物理数据块的映射关系，用于重构完整文件。
2. 密钥元数据记录了文件内逻辑数据块的解密密钥，用于恢复相应的明文内容，由于密钥元数据包含密钥信息，需由文件属主的主密钥（master key）加密后以密文形式存储。

1.1.2 问题和动机

数据块频率泄漏问题

由于基于数据块内容产生密钥，加密重复数据删除泄漏了数据块的频率信息，即如果一个明文数据块出现了 n 次，则它对应的密文数据块也将出现 n 次。另一方面，真实数据集中数据块的出现频率往往呈非均匀分布，本文调研了 FSL 和 VM 备份数据集 (数据集信息见 §3.2) 的数据块频率分布特征，发现三种数据集有超过 97% 的数据块的频率低于 100 次，而至多只有 0.04% 的数据块的频率高于 10,000 次（图1-3），这种非均匀的分布特点使攻击者可以利用频率来确定相应数据块。

基于以上原因，认为加密重复数据删除可能受到频率分析[12]的威胁，拟通过本课题，深入研究频率分析攻击对加密重复数据删除安全性的影响，以及提高频率分析攻击效果的方法。

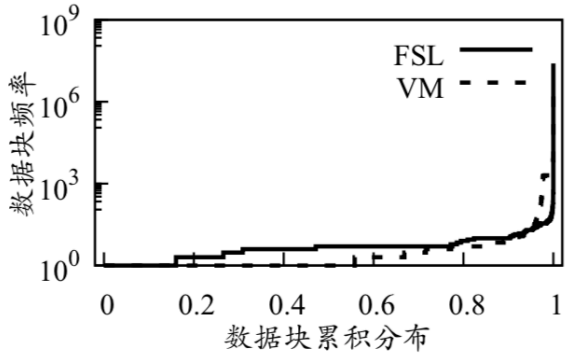


图 1-3 两种真实数据集的数据块频率分布

1.1.3 研究意义

本课题研究将填补频率分析攻击研究空白，对理解加密重复数据删除的实际安全性，并降低其在非适合场景下的误用风险具有重要作用。

尽管加密重复数据删除的频率泄漏已是学术界公认的安全问题，但针对性的频率分析攻击研究仍为空白 (即利用频率泄漏来获取隐私数据仍是一个开放性问题)，致使部分厂商盲目地将加密重复数据删除技术应用于商业产品[13,14]和开源系统[15–18]中。本项目将研究加密重复数据删除技术在频率分析攻击下的实际安全性，以指导其在适合场景下被正确使用。

1.2 国内外研究历史与现状

1.2.1 加密重复数据删除

在传统对称加密方式下，每个用户具有不同的密钥，不同用户之间的相同明文会被加密为不同密文，难以执行 (密文) 重复数据删除操作。

消息锁定加密（message-locked encryption，MLE）确立了加密重复数据删除的密码学基础[10]: 基于数据内容产生密钥 (称为 MLE 密钥)，从而将相同明文加密为相同密文。最流行的 MLE 实例是收敛加密（convergent encryption，CE）[11]，它使用明文的散列值作为 MLE 密钥，并基于密文散列值计算指纹，以识别重复数据（如图1-2）。基于 CE 的加密重复数据删除方案还包括:

1. 散列收敛加密（hash convergent encryption，HCE）[11]与 CE 具有相同的 MLE 密钥产生规则，但基于明文散列值计算指纹。

2. 随机收敛加密（random convergent encryption，RCE）[11] 使用随机密钥加密以产生非确定的密文，同时也基于明文散列值来进行重复检查。
3. 收敛扩散（convergent dispersal，CD）[19] 使用明文散列值作为秘密共享（secret sharing）的输入种子，在兼容重复数据删除的基础上提高了密文存储的可靠性。

上述 MLE 实例基于明文产生 MLE 密钥（CE、HCE 和 CD）或指纹（HCE 和 RCE），如果明文是可预测的 (即所有可能的明文的数量有限)，这些方案易于受到离线暴力破解攻击[10,20]。为了抵御该攻击，DupLESS[20] 基于第三方密钥服务器实现了服务器辅助 MLE（server-aided MLE），确保无法从离线明文派生出相应的 MLE 密钥。以服务器辅助 MLE 为基础，现有研究进一步解决了加密重复数据删除的故障容错[21,22]、透明价格模型[23]、点对点密钥管理[24]、层次密钥管理[25] 等问题。围绕 MLE 扩展功能的一系列研究包括: 兼容加密重复数据删除的数据完整性审计协议[26]；支持动态访问控制的加密重复数据删除系统 REED[27,28]。

无论是 MLE 还是服务器辅助 MLE，均须为相同的明文产生相同的密文（CE、HCE、CD 和 DupLESS）或指纹（HCE 和 RCE），泄漏了数据的出现频率。一些理论研究[29–31] 基于零知识证明、全同态加密、双线性对运算等底层密码技术实现了随机加密，但这些方案存在计算复杂性高、依赖多轮信息交互等问题，难以应用到系统实践中。本文关注可实际应用的加密重复数据删除系统/方案的频率泄漏问题，研究其安全性影响和防御对策。

1.2.2 频率分析攻击

频率分析[32] 是一种针对确定性加密（deterministic encryption）的密码分析技术，被应用于破解匿名查询日志[33]、破坏关键词隐私[34–38]、重构密文数据库记录[12,39–43] 等实际攻击。本项目研究针对数据块的频率分析攻击，与现有攻击目标（查询日志条目、关键词、数据库记录等）相比，数据块数量极其庞大（呈千万级），并且大量数据块具有相同频率，致使当前的频率分析攻击算法难以适用。

面向加密重复数据删除，已有工作[44] 提出了基于数据块局部性（chunk locality）[45–47] 的频率分析攻击。本课题的部分技术路线（§ 1.4）就是在该工作[44] 的基础上提出来的，并进一步研究频率分析攻击的准确率和依赖条件，以及面向真实系统的攻击原型。除了频率分析和暴力破解（§ 1.2.1）之外，加密重复数据删除还可能遭受边信道攻击[9,48,49]、副本伪造攻击[10]、基于数据块长度的攻击[50] 等威胁，但这些攻击可通过所有权证明[48]、守卫解密 (guarded decryption)[10]、固定长度分块等措施进行防御，而本文所研究的频率分析攻击超出了现有保护措施的

防御范畴。

1.3 课题的研究内容、研究目标、以及拟解决的关键问题

1.3.1 研究内容

加密重复数据删除的频率分析攻击

在传统频率分析模式下，攻击者能够访问明文逻辑数据块集合 M 和密文逻辑数据块集合 C（M 和 C 包含重复的明文和密文数据块）。攻击者根据出现频率分别对 M 和 C 中的数据块进行排序，然后将 C 中的密文数据块映射为 M 中与其具有相同排名的明文数据块。但是，传统频率分析在加密重复数据删除中难以形成有效的攻击，主要原因是：M 和 C 的原始内容可能存在差异 (例如 M 和 C 来源于同一个文件系统在不同时间点的备份镜像)，将打乱数据块频率排序的对应关系；并且，在频率排序过程中可能存在大量明文和密文数据块具有相同的频率，频率分析难以排序这些数据块来形成正确的对应关系。

为了提高传统频率分析的攻击效果，首先研究基于数据特征的新型频率分析攻击技术。然后，分别从抵抗频率排序干扰和降低攻击发生条件两方面改进攻击技术。最后，实现针对真实系统的频率分析攻击原型，并分析该攻击对各类数据安全性的影响。

1.3.2 研究目标

针对以上研究内容，预期实现如下研究目标:

1. 在理论上，构造针对加密重复数据删除的频率分析攻击，揭示实践中的安全隐患。
2. 在技术上，以理论研究为支撑，设计并实现针对加密重复数据删除系统/方案的频率分析攻击工具，并在真实系统中进行理论验证和攻击效果测试。

1.3.3 拟解决的关键问题

本课题致力于解决传统频率分析攻击方法在针对加密重复数据删除方案/系统进行攻击时难以解决的如下问题:

1. 明密文数据块的的原始内容可能存在差异 (例如一组明密文集合分别来源于同一个文件系统在不同时间点的备份镜像)，将打乱数据块频率排序的对应关系。
2. 在频率排序过程中可能存在大量明文和密文数据块具有相同的频率，基本频率分析攻击难以排序这些数据块来形成正确的对应关系。

1.4 拟采取的研究方案

如图1-4所示，在支撑研究（基于数据块局部性的频率分析攻击方案[44]）的基础上，根据相对频率分布特性，设计抗排序干扰的攻击方法；在数据相似性的基础上，设计低依赖条件的攻击方法。最终设计出针对加密重复数据删除方案/系统的新型频率分析攻击方法及对应的原型软件工具。

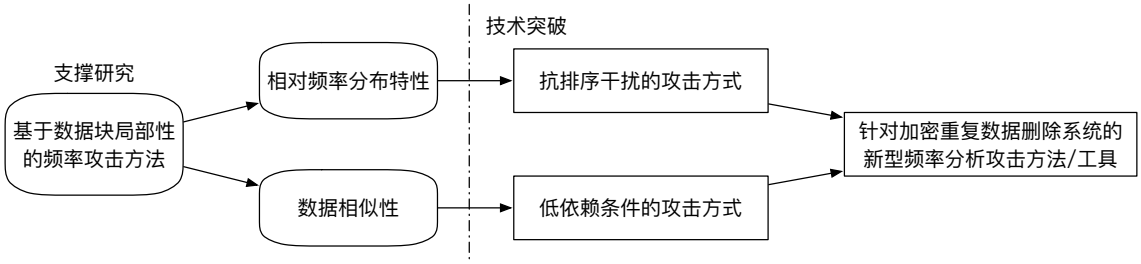


图 1-4 技术路线图

1.4.1 支撑研究

基于数据块局部性的频率分析攻击方案（locality-based attack）[44] 作为支撑研究，该方案主要用于破译加密数据备份，即已知的明文数据块集合 M 和目标密文数据块集合 C 源于同一个系统在两个不同时间点的备份镜像。攻击利用了数据块的局部性特征：在不同备份之间，绝大多数数据块保持了相同的局部顺序；例如，每天备份工作项目的进度快照，若一天内的改动较小，则在两次备份之间未被改动的大部分数据块之间的相对顺序保持不变。因此，得出一个关键推论：如果明文数据块 M 是密文数据块 C 的原始明文，那么 M 左边和右边相邻的明文数据块有较大可能也是 C 左边和右边相邻密文数据块的原始明文。

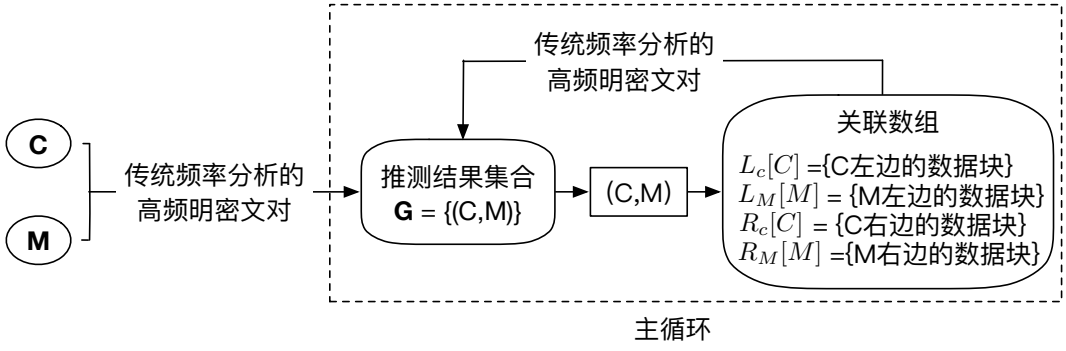


图 1-5 基于数据块局部性的频率分析攻击

基于此，攻击流程如图1-5所示：首先，对 C 和 M 应用传统频率分析（§ 1.2.2），将获得的若干组高频明密文对加入推测结果集合 G；然后，每次从 G 中选取一组

明密文对 (M, C)，分别对其左右相邻的明文和密文数据块集合 $L_M[M]$ 和 $L_C[C]$，以及 $R_M[M]$ 和 $R_C[C]$ 实施频率分析，并将获得的高频明密文对也加入 G；继续对 G 中的明密文对进行基于相邻数据块的频率分析，直至所有明密文对都被处理。

为了验证攻击效果，定义推测率为正确推测出原始明文的 (不同) 密文数据块个数与 C 中 (不同) 密文数据块总个数的比率。在基于真实数据集的实验验证中，攻击方案能够达到 17.8% 推测率，远远高于传统频率分析方法的 0.0001% 推测率。

1.4.2 技术突破

1.4.2.1 基于分布的频率分析攻击方法

基于分布的频率分析攻击利用密文和明文的相对顺序信息来增强频率分析的有效性。该攻击方法建立在数据块局部性 [45–47] 的基础上。通过以下三个关键特性构造出基于分布的频率分析攻击方法。

- 数据块局部性指出数据的原始排序可能会在各种备份文件中得以保留，因此可以用于在备份文件中推断出类似的与位置相关的密文-明文对。
- 相邻数据的共现频率的相对频率分布可以通过分析得到。
- 明文及其相应的密文具有相似的相对频率分布的特性。

1.4.2.2 基于聚类的频率分析攻击方法

在基于分布的频率分析攻击方法的基础上，通过引入相似性 [51] 这一属性来消除基于分布的频率分析攻击方法对明文数据的细粒度顺序信息的需求。基于此提出基于聚类的频率分析攻击方法。

1.5 可行性分析

1.5.1 研究方法可行

本课题通过引入加密重复数据删除方案/系统的相关特性来设计新型频率分析攻击方法，针对性提高频率分析攻击方法的有效性。该研究问题具有很高的实际价值且在面向加密重复数据删除的攻击中，已有工作提出了基于数据块局部性（chunk locality） [45–47] 的频率分析攻击 [44]，在此基础上改进攻击方案以提高频率分析攻击实际效果具有明确的可行性。

1.5.2 研究条件可行

前期基于数据块局部性的频率分析攻击方案给出了本课题研究两种新型频率分析攻击方法的基本理论和工具基础。在原有方案上改进即可用于本课题研究。

研究者在本科阶段深入钻研了 CDStore、REED、SDFS 等加密重复数据删除系统，以及前期基于数据块局部性的频率分析攻击方案的理论和工具实现。这些经验可以帮助设计新型频率分析方法以及其在真实系统中的实践。

1.6 本文特色与创新之处

本文的特色与创新之处有以下几点：

1. 本文针对加密重复数据删除提出了两种新型的频率分析攻击方法。除了利用由于加密重复数据删除具有确定性导致的频率泄漏之外，两种攻击都利用重复数据删除工作负载的特性来增加频率分析攻击的有效性。
2. 本文使用多个真实数据集（包括长期备份[52,53]，Windows 文件系统快照[54]和 VM 磁盘映像[19,28]）以及开源重复数据删除系统 SDFS[55]、Destor[56]，评估两种新型频率分析攻击方法和基本频率分析攻击方法的实际效果。本文通过评估频率攻击方法的攻击结果，进一步分析本频率分析攻击方法对实际加密重复数据删除带来的安全性影响。
3. 本文讨论了减少加密重复数据删除中信息泄漏的可能方案。

1.7 论文结构及安排

第二章 预备知识

在本章中，将介绍普通和加密重复数据删除方法的基础知识，以及加密重复数据删除仍然容易受到攻击的原因。

2.1 重复数据删除

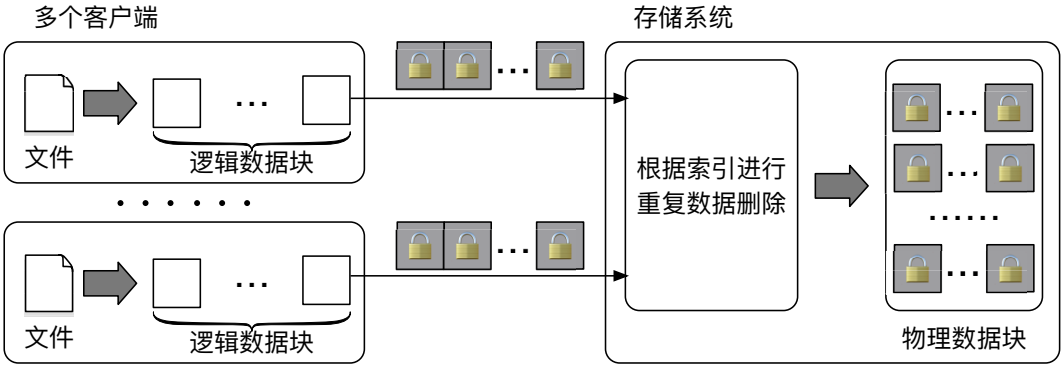


图 2-1 重复数据删除的工作流程概览

本文专注于基于块的重复数据删除，它以称为数据块的小型数据单元为粒度运行。图2-1总结了重复数据删除工作流程。

具体地，重复数据删除系统首先通过文件切块过程将客户端的文件（例如，备份文件）分割成逻辑数据块，根据每一个逻辑数据块的内容使用 HASH 算法（散列算法）计算得到其对应的唯一标签（又成为指纹）。如果两个数据块具有相同的标签，那么认为两个数据块相同（不同逻辑数据块计算得到相同的标志的概率忽略不计[57]），即数据块内容和其标志完全一致；若两个逻辑数据块标签不一致，显然两逻辑数据块不同。重复数据删除系统的存储系统仅存储相同逻辑数据块的副本（称为物理数据块），并且每个相同的逻辑数据块通过一个较小空间开销的索引引用相同的物理数据块。

基于块的重复数据删除比基于文件的重复数据删除更精细，因此通常具有更高的存储效率（节省更多的存储空间）。基于块的重复数据删除有两种基本的数据块分块方法：固定大小的数据分块方法和可变大小的数据分块方法。固定大小的数据分块方法将数据分割为大小相同的逻辑数据块，而可变大小的数据分块方法（也称为内容定义的数据块分块方法）通常以内容相关的方式指定数据块的边界（例如，通过 Rabin 指纹识别[58]）将文件数据分区为可变大小的数据块。固定大小的分块方法简单而快速，而可变大小的分块方法可以在内容发生变换时保持

存储效率，因为即使文件添加或删除了某些内容，大多数数据块仍保持不变。可变大小的分块通常用在备份系统中（例如，[45,46]），但固定大小的分块显然对某些工作负载更加有效（例如，VM 备份数据集[59]）。本文的工作同时涉及固定大小和可变大小的分块方法产生的数据块。

加密的重复数据删除解决了外包环境（例如，云存储）中的数据块机密性，同时保持了重复数据删除的有效性。出于安全考虑，用户希望在将自己的数据外包之前对其数据进行加密，以确保个人数据的隐私性。传统的对称加密要求多个客户端通过其（不同的）密钥加密其文件数据，即使相同的明文数据块也会被加密为不同的密文数据块，服务器无法感知到这些密文数据块所对应的的原始明文数据块内容是否相同。消息锁定加密（MLE）[10] 给加密重复数据删除提供了新的处理方法。最基本的 MLE 实例化是基于每个明文的内容（例如，基于块的重复数据删除中的逻辑数据块）导出其对称密钥（称为 MLE 密钥），并使用 MLE 密钥加密明文以形成对应的密文（例如，加密的逻辑数据块）。因此，MLE 可以确保将相同的明文加密为相同的密文。最后，存储系统从每个密文中导出其对应的标签并执行重复数据删除。

2.2 重复数据删除中存在的威胁

2.2.1 MLE 加密具有确定性

MLE 可以实现为不同的实例化（参见章节：第九章）。其中最受欢迎的是融合加密（CE）[11]，它已经在各种存储系统中实现[13–17,60–63]。CE 的主要思想是根据每个明文数据块的内容通过 HASH 函数（散列函数）导出其对应的 MLE 密钥。这种方法可以确保具有相同内容的逻辑数据块加密得到的密文数据块始终一致，即可使用密文数据块的标签进行重复数据删除。但也存在部分 MLE 实例[10] 允许将相同的明文加密成不同的密文。当然，他们确保用于重复数据删除的标签仍然来自明文，因此他们依然可以通过检查标签的相同性来执行重复数据删除。

现有的 MLE 实例都建立在确定性加密这一概念的基础之上，确定性加密可以确保密文（或标签）从明文确定性地导出，与传统的对称加密相比，可以保留重复数据删除的有效性。虽然确定性加密提供了机密性保证，但是本文认为攻击者可以利用这种确定性来推断给定密文的原始明文。

2.2.2 理论上加密重复数据删除存在信息泄漏

实际的加密重复数据删除系统会暴露一些关于原始块的信息，称为泄漏。最基本的泄漏是数据块的频率，这一点已在先前的工作中得到承认[29,31,44,50]。具体

地说，MLE 会将相同的明文数据块加密成相同的密文数据块，这一过程泄漏了每个原始明文数据块出现在输入数据中的次数。本文认为频率泄漏难以避免，因为这会破坏明文和密文数据块的一对一映射关系，进而降低重复数据删除系统的性能[29]或存储效率[44]。

已有的研究工作已经探索了实际加密重复数据删除系统存在的的其他类型的泄漏[44,50]。首先，一些重复数据删除系统[45,47]为了实现更高的性能，需要按照数据块在原始文件中出现的顺序来操作数据块；这暴露了铭文数据块的逻辑顺序，基于该逻辑顺序可以识别输入文件中每个数据块的位置。其次，一些工作[11,20,64]为了降低存储开销，不会填充任何冗余的密文数据块，这暴露了原始明文数据块的数量信息，因为密文数据块应该与原始明文数据块具有相同的数量。

在已有工作[44,50]中，通过构建攻击原语来推断原始明文数据块[44]或根据特定文件的存在[50]来攻击加密重复数据删除方案。但现有的其他关于加密重复数据删除的工作仍未探索过如何基于推断得到的信息发起实际攻击。本文对加密重复数据删除的泄漏问题进行了全面研究，以求回答以下两个问题:

1. 如何在各种泄漏中通过启用基元来推断原始内容?
2. 以推断内容为基础的实际攻击的具体含义是什么?

2.2.3 观察得到的实际系统中的泄漏

下面将讨论在最先进的加密重复数据删除方案/系统中观察到的各种泄漏。

- **频率泄漏**

  加密的重复数据删除方案应用了确定性加密（例如，MLE），其中相同的明文数据块将被加密成相同的密文数据块。这泄漏了每个数据块的频率信息（即原始数据块出现在输入数据中的次数）。频率泄漏是 MLE 难以解决的问题，因为它很难被预防（否则，将导致重复数据删除性能显着下降[29]或存储空间开销大幅提高[44]）。

- **数据块长度信息泄漏**

  为了减轻存储开销，建议从业者在没有填充方案[50]的情况下实现块加密。例如，Farsite[11]，Tahoe-LAFS[64]和 DupLESS[20]在计数器模式下使用 AES 加密，但它们不会使用任何其他数据填充密文数据块。因为密文数据块应该与原始明文数据块具有相同的大小，这会泄漏原始数据块大小信息[50]。

  对于可变大小的数据块，这种泄漏更加严重，由于数据块边界通过匹配特定内容的方法来识别（例如，通过 Rabin 指纹识别[58]）。具有不同大小的两个密文数据块进一步暴露了它们来自于不同明文数据块的信息。

- **数据块逻辑顺序泄漏**

  在部署加密重复数据删除时无意中发生泄漏，使得数据块逻辑顺序被外部所获的。

2.3 频率分析攻击

频率分析已在一些已有的工作[12,34,35,35–40,42,43]中用来构建攻击。例如针对数据库中的结构化关系数据的攻击，通过经典频率分析[12]，探索表列的相关性或排序。

本文旨在应用频率分析来探索加密重复数据删除的漏洞。我们的重点不是解决如何获得用于推理攻击的相关明文。在统频率分析模式下，攻击者能够访问明文逻辑数据块集合 M 和密文逻辑数据块集合 C（M 和 C 包含重复的明文和密文数据块）。攻击者根据出现频率分别对 M 和 C 中的数据块进行排序，然后将 C 中的密文数据块映射为 M 中与其具有相同排名的明文数据块。

2.4 本章小结

本章介绍了普通与加密重复数据删除，现有加密重复数据删除中存在的信息泄漏，以及频率分析攻击的典型应用和其在加密重复数据删除中的基本运用方式。

第三章 威胁模型分析

In this section, we formulate the threat model for our proposed frequency analysis attacks against encrypted deduplication.

3.1 Definitions

We first provide the definitions for our threat model. We model a plain file as an ordered list of *n logical plaintexts* (i.e., logical chunks before deduplication), denoted by $\mathbf{M} = \langle \hat{M}^{(1)}, \hat{M}^{(2)}, \ldots, \hat{M}^{(n)} \rangle$. Each logical plaintext $\hat{M}^{(i)}$ (where $1 \leq i \leq n$) is encrypted via MLE into the corresponding ciphertext $\hat{C}^{(i)}$, and all encrypted results of $\mathbf{M}$ form a stream of *n logical ciphertexts* $\mathbf{C} = \langle \hat{C}^{(1)}, \hat{C}^{(2)}, \ldots, \hat{C}^{(n)} \rangle$. Note that the logical ciphertexts in $\mathbf{C}$ are also ordered to reflect the deduplication processing sequence of an encrypted deduplication storage system.

Identical logical plaintexts and ciphertexts may appear at different positions of $\mathbf{M}$ and $\mathbf{C}$, respectively. We denote a unique plaintext as M (uniquely identified by its tag), which is encrypted via MLE to the corresponding unique ciphertext C. Each M (resp. C) corresponds to one or multiple identical copies of $\hat{M}^{(i)}$ (resp. $\hat{C}^{(i)}$).

3.2 Adversarial Goals and Assumptions

We consider an adversary that aims to infer a set of ciphertext-plaintext pairs, denoted by $\{(C, M)\}$, with two specific goals:

- **High inference rate:** A large fraction of correct ciphertext-plaintext pairs are inferred, among all correct ciphertext-plaintext pairs (i.e., high recall or low false negative rates in statistical terms).
- **High inference precision:** A large fraction of ciphertext-plaintext pairs are correct, among all inferred ciphertext-plaintext pairs (i.e., high precision or low false positive rates in statistical terms).

We assume that the adversary is *honest-but-curious*, such that it can passively monitor the stream of ciphertexts $\mathbf{C}$ being written to the storage system and exploit different types of leakage information from $\mathbf{C}$ (see Section **??**). Given the available leakage information, the adversary aims to infer the original plaintext of each ciphertext in $\mathbf{C}$ via a *ciphertext-only* attack. It is possible that the adversary knows a limited subset of ciphertext-plaintext

pairs to launch a known-plaintext attack, which further strengthens attack severity [**?**]. We do not consider the known-plaintext attack in this paper.

We assume that the adversary does not have access to any *metadata* that contains the information about how chunks are operated and stored, as we do not apply deduplication to the metadata and hence it can be protected by traditional symmetric encryption. Also, we assume that the adversary does not have *active* capabilities, which can be independently prevented by existing approaches. One example is that a malicious client may claim the ownership of unauthorized files in client-side deduplication [**?**, **?**, **?**]; it can be addressed by proof-of-ownership [**?**, **?**, **?**] or server-side deduplication [**?**, **?**]. Another example is that a malicious storage system may modify stored data; it can be detected by remote integrity checking [**?**, **?**].

3.3 Leakage Channels

We consider three types of leakage channels in encrypted deduplication storage that enable an adversary to infer information:

- **Frequency:** Due to the deterministic nature of encrypted deduplication, the frequency of each ciphertext in **C** before deduplication (i.e., the number of duplicate copies) can be mapped to that of its corresponding plaintext in **M**.
- **Order:** Some storage systems (e.g., [**?**, **?**, **?**]) apply deduplication to the chunks in the same order as they appear in the original file. Thus, the order of ciphertexts in **C** can be mapped to that of plaintexts in **M**.
- **Size:** Variable-size chunking (see Section 第二章) leads to different chunk sizes of the ciphertexts. If such ciphertexts are not padded (to avoid storage overhead [**?**]), the size of a ciphertext in **C** can be mapped to that of its corresponding plaintext in **M**.

In addition, the adversary has access to some *auxiliary information* that presents the ground truth about the data characteristics correlated with **M**. Note that the availability of auxiliary information is necessary for any inference attack [**?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**]. In this work, we consider the auxiliary information as an ordered list of previously known plaintexts (e.g., old user backups or VM disk images), denoted by **A**. Clearly, the attack severity depends on the correlation between **A** (i.e., the previously known plaintexts) and **M** (i.e., the plaintexts that are to be inferred). Our focus is not to address how an adversary can obtain auxiliary information, possibly due to careless data release [**?**], stolen storage devices [**?**], and cloud leakage [**?**]. Instead, given such information, we investigate how the

available auxiliary information, combined with the leakage channels, bring information leakage from encrypted deduplication.

We focus on frequency analysis [**?**] as the attack methodology. Classical frequency analysis sorts the unique ciphertexts in **C** and the unique plaintexts in **A** by *frequency* (i.e., the number of identical copies corresponding to each unique ciphertext or unique plaintext). It then relates each unique ciphertext in **C** with the unique plaintext in **A** that has the same frequency rank. In the following sections, we design sophisticated frequency analysis attacks against encrypted deduplication.

第四章 基于分布的频率分析攻击方法

The distribution-based attack exploits the order information of **C** and **A** to strengthen the effectiveness of frequency analysis. It then compares the relative frequency distributions of **C** and **A** to reduce the false positive results. We also show how to exploit chunk size information to further improve the inference precision.

4.1 Background: Locality-based Attack

The distribution-based attack builds on the previously proposed *locality-based attack* [**?**], which shows how frequency analysis causes information leakage in backup workloads. The locality-based attack considers *full backups* (or backups for short) that are created as the complete copies of primary data (e.g., application states, file system snapshots, and VM disk images) on a daily or weekly basis. It aims to infer the ciphertext-plaintext pairs across different versions of backups. It assumes that the auxiliary information **A** is derived from some older backups, and its goal is to infer the plaintexts of the latest backup (i.e., **M**).

The locality-based attack leverages the *locality* property [**?**, **?**, **?**], which is a common phenomenon in practical backup workloads. Specifically, the locality property states that neighboring chunks tend to co-occur in the *same order* across different versions of backups before deduplication. The main reason is that the updates to each backup are often clustered in some small regions of chunks, while the remaining large stretch of chunks appear unchanged or preserve the same order across different versions of backups.

Based on locality, the locality-based attack exploits the order information to discover the neighboring information of ciphertexts and plaintexts. Specifically, for a given unique ciphertext C, the attack first identifies the set of all identical copies $\{\hat{C}^{(i)}\}$. For each $\hat{C}^{(i)}$, it considers the left and right *neighbors* of $\hat{C}^{(i)}$, i.e., $\hat{C}^{(i-1)}$ and $\hat{C}^{(i+1)}$, respectively. It extracts the sets of left and right neighbors into the associative arrays $\mathbf{L_C}$ and $\mathbf{R_C}$, respectively. The associative arrays store the mappings of each unique ciphertext C and its *co-occurrence frequencies* with its left and right neighbors, respectively. Similarly, the attack also constructs the associative arrays $\mathbf{L_A}$ and $\mathbf{R_A}$ based on the order information of **A**.

The locality-based attack then iterates frequency analysis through the neighbors of each inferred ciphertext-plaintext pair. It first applies frequency analysis to infer a number

(parameterized by u) of top-frequent ciphertext-plaintext pairs $\{(C, M)\}$ from **C** and **A**. The inferred results are likely to be *real* (i.e., the target ciphertext is indeed mapped from the inferred plaintext), based on the observation that the ranks of highly frequent chunks are stable across different versions of backups. For each inferred pair (C, M), the attack finds their left and right neighbors that have the most co-occurrence frequencies with C and M, respectively. Due to locality, the left and right neighbors of M are likely to be the original plaintexts of the corresponding left and right neighbors of C, respectively. Thus, the attack also includes the top-frequent left (resp. right) neighbors of C and M into the set of the resulting inferred ciphertext-plaintext pairs. Finally, the attack procedure iterates until the neighbors of each inferred ciphertext-plaintext pair are examined.

However, the locality-based attack has a major weakness that it introduces a high number of false positives (i.e., incorrect ciphertext-plaintext pairs). Since the main idea of frequency analysis is to map ciphertexts to plaintexts with the same frequency ranks, any disturbance to frequency ranking (e.g., the updates across backups) can lead to incorrect ciphertext-plaintext pairs, which can in turn comprise the inference of their neighbors. Although the locality-based attack is shown to effectively infer a significant fraction of real ciphertext-plaintext pairs, the adversary has low confidence to tell whether each inferred ciphertext-plaintext pair is real or a false positive.

4.2 Description of Distribution-based Attack

The distribution-based attack extends the locality-based attack [**?**] to significantly remove false positives. It leverages the locality property in backup workloads as in the locality-based attack. In addition, for each unique ciphertext C in **C**, we measure its *relative frequency distribution* based on its co-occurrence frequencies with its neighbors. Similarly, we examine the relative frequency distribution of each unique plaintext M in **A**. Our observation is that for a real ciphertext-plaintext pair (C, M), both C and M should have similar relative frequency distributions; i.e., their co-occurrence frequencies with their respective neighbors are similar. We treat this observation as a more general notion of the locality property, and use it as a condition to filter possibly incorrect ciphertext-plaintext pairs.

Figure 4-1 presents the workflow of the distribution-based attack. We first sort the unique ciphertexts and plaintexts by their frequencies in **C** and **A**, respectively. As in the locality-based attack [**?**], we configure the parameter u and the underlying frequency

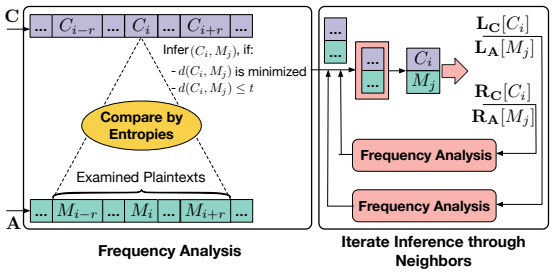


图 4-1 Workflow of distribution-based attack.

analysis to return at most u ciphertext-plaintext pairs. In particular, for each unique ciphertext C_i of rank i (where $1 \le i \le u$), we examine a number of unique plaintexts $M_{i-r}, \ldots, M_i, \ldots, M_{i+r}$ that rank from $i-r$ to $i+r$, where r is a configurable parameter (e.g., 10 by default) that indicates the maximum range of rank disturbance that can be addressed.

For each C_i (where $1 \le i \le u$) and the corresponding M_j (where $i-r \le j \le i+r$), we compare their relative frequency distributions by *entropy*, a key concept in information theory that measures the amount of information produced by a random source. Here, we adopt the entropy to *quantify the randomness of probability distribution* [**?**]. Specifically, we define two random variables, denoted by X and Y, to describe the co-occurrences of C_i with its left and right neighbors, respectively, such that the event "$X = C$" denotes the case that C is the left neighbor of C_i, and the event "$Y = C$" denotes the case that C is the right neighbor of C_i. Thus, we compute the probabilities of both events based on $\mathbf{L_C}$ and $\mathbf{R_C}$:

$$\Pr[X = C] = \frac{\mathbf{L_C}[C_i][C]}{\sum_{C' \in \mathbf{L_C}[C_i]} \mathbf{L_C}[C_i][C']},$$
$$\Pr[Y = C] = \frac{\mathbf{R_C}[C_i][C]}{\sum_{C' \in \mathbf{R_C}[C_i]} \mathbf{R_C}[C_i][C']},$$

where $\mathbf{L_C}[C_i]$ and $\mathbf{R_C}[C_i]$ store the left and right neighbors of C_i, respectively, while $\mathbf{L_C}[C_i][C']$ and $\mathbf{R_C}[C_i][C']$ store the co-occurrence frequencies of C_i with its left and right neighbor C'. Both X and Y follow the relative frequency distributions of C_i, and we characterize their randomness by entropies denoted by $e(\mathbf{L_C}[C_i])$ and $e(\mathbf{R_C}[C_i])$, respectively:

$$e(\mathbf{L_C}[C_i]) = \sum_{C \in \mathbf{L_C}[C_i]} \log_2 \frac{1}{\Pr[X = C]},$$
$$e(\mathbf{R_C}[C_i]) = \sum_{C \in \mathbf{R_C}[C_i]} \log_2 \frac{1}{\Pr[Y = C]}.$$

Similarly, we compute the entropies $e(\mathbf{L_A}[M_j])$ and $e(\mathbf{R_A}[M_j])$ of each M_j based on $\mathbf{L_A}$ and $\mathbf{R_A}$, respectively. We then quantify the similarity of the relative frequency distributions of C_i and M_j via the *Euclidean distance*, denoted by $d(C_i, M_j)$:

$$\begin{aligned} d(C_i, M_j) &= \Big\{[e(\mathbf{L_C}[C_i]) - e(\mathbf{L_A}[M_j])]^2 \\ &\quad + [e(\mathbf{R_C}[C_i]) - e(\mathbf{R_A}[M_j])]^2\Big\}^{1/2}. \end{aligned}$$

Clearly, C_i and M_j have similar relative frequency distributions only when the Euclidean distance $d(C_i, M_j)$ of their entropies is small. Thus, we identify (C_i, M_j) as an inferred ciphertext-plaintext pair if they satisfy the following requirements:

- **R1:** $d(C_i, M_j)$ is the *smallest* for all $i - r \leq j \leq i + r$.
- **R2:** $d(C_i, M_j)$ is not larger than a pre-defined parameter t (e.g., 1 by default).

One special note is that the original plaintext of C_i may fall outside of the examined plaintexts $M_{i-r}, \ldots, M_{i+r}$. In this case, R1 is still satisfied by some M_j ($i - r \leq j \leq i + r$), and we expect to filter the incorrect ciphertext-plaintext pair by R2.

Then, we adopt the iteration paradigm in the prior locality-based attack [**?**] (see Section 4.1) to increase the coverage of inferred ciphertext-plaintext pairs. Specifically, for each inferred (C_i, M_j), we apply the new frequency analysis scheme (see above) to infer more ciphertext-plaintext pairs through the neighbors of C_i and M_j, and further iterate the same inference for those newly inferred pairs. We finally stop the iteration, when none of new ciphertext-plaintext pairs can be inferred.

4.2.1 Summary

To summarize, the distribution-based attack provides a more general notion of locality by considering the relative frequency distribution during frequency analysis. It is configured by three parameters (i) u, which specifies the maximum number of ciphertext-plaintext pairs returned by frequency analysis, (ii) r, which specifies the maximum range of rank disturbance to be considered, and (iii) t, which specifies the Euclidean distance threshold to filter possibly incorrect inference results. The prior locality-based attack [**?**] can be regarded as a special case of the distribution-based attack under the parameter configuration of $r = 0$ (i.e., without addressing any disturbance to frequency ranking) and $t \to \infty$ (i.e., without filtering any incorrect inference results).

4.3 Exploiting Size Leakage

We propose an advanced variant of the distribution-based attack that operates with size information to further reduce false positives. Specifically, we assume that the size of each ciphertext in $\mathbf{C}$ reflects that of its original plaintext.

Our idea builds on the fundamental truth that if a ciphertext C corresponds to a plaintext M, then the size of C approximates that of M. This is because encrypted deduplication preserves the number of blocks (i.e., the basic units operated by symmetric encryption) in the content to be encrypted. We use this fact to further filter the incorrect ciphertext-plaintext pairs (C, M), where the number of blocks in C and M are different.

In this work, we assume that each block is of 16 bytes, which is a typical configuration for AES. For each considered ciphertext-plaintext pair (C_i, M_j), we derive the number of blocks in C_i and M_j as $b(C_i)$ and $b(M_j)$, respectively:

$$b(C_i) = \lceil \frac{\mathsf{size}(C_i)}{16} \rceil,$$
$$b(M_j) = \lceil \frac{\mathsf{size}(M_j)}{16} \rceil,$$

where $\mathsf{size}(C_i)$ and $\mathsf{size}(M_j)$ are the exact sizes of C_i and M_j, respecitively, and $\lceil x \rceil$ returns the smallest integer greater than or equal to x. In addition to R1 and R2 (see Section 4.2), we apply the following requirement to filter by size:

- **R3:** $b(C_i)$ equals $b(M_j)$.

The requirement R3 is effective to filter the incorrect ciphertext-plaintext pairs regarding variable-size chunks, where different chunks possibly have distinct sizes. However, for fixed-size chunks, it is always satisfied by the ciphertext-plaintext pairs, even they are incorret. Despite of this, we can still apply R1 and R2 to achieve high-precision attack.

第五章 基于聚类的频率分析攻击方法

In this section, we relax the adversarial knowledge in the distribution-based attack, and propose the *clustering-based attack*, which does not need the fine-grained ordering of plaintexts. Instead, it exploits the *similarity* property to infer original data from similar *clusters* (i.e., some large data units aggregated by chunks), without relying on the ordering of chunks in each cluster. We first introduce similarity, and then show how to adapt this property into inference attack.

5.1 Background: Similarity

Similarity [**?**] states that backup files from the same source are likely to be *similar* and share a large fraction of identical chunks. The similarity between backup files can be quantified by *Broder's theorem* [**?**]. Specifically, if we treat each file as a set S of chunks (i.e., ignore their orders), Broder's theorem states that if the probability that two sets of chunks share the same minimum chunk hash is high, then both sets are likely to share a large fraction of identical chunks and vice versa:

$$\Pr[\min\{\mathrm{H}(S)\} = \min\{\mathrm{H}(S')\}] = \frac{|S \cap S'|}{|S \cup S'|},$$

where $\mathrm{H}(\cdot)$ is a hash function that is chosen uniformly at random from a min-wise independent family of permutations, and $\min\{\mathrm{H}(S)\}$ returns the minimum chunk hash of S.

Prior works that target different aspects (e.g., performance [**?**, **?**, **?**], and security [**?**]) of deduplication have applied similarity to *preserve storage efficiency*. Specifically, they operate deduplication just on the files that share the same minimum chunk hash (i.e., similar); since similar files can have a large number of identical chunks due to Broder's theorem, such *near-exact* deduplication only leads to a slight degradation of storage efficiency. Different from prior approaches [**?**, **?**, **?**, **?**], we apply similarity to increase the severity of frequency analysis.

5.2 Description of Clustering-based Attack

We now present the clustering-based attack (Figure 5-1) that builds on similarity to infer original data against encrypted deduplication.

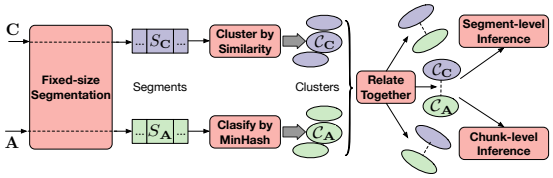


图 5-1 Workflow of clustering-based attack: MinHash denotes the minimum chunk hash of a plaintext segment.

To exploit similarity, we first introduce *segmentation* on a stream of chunks. Specifically, we partition **C** into a set of coarse-grained data units, called *ciphertext segments*. Each ciphertext segment, denoted by $S_{\mathbf{C}}$, comprises multiple adjacent ciphertexts in **C**. In this work, we implement the *fixed-size* segmentation scheme to ensure that all ciphertext segments are of the same size (e.g., 4MB by default). For compatibility, the same fixed-size segmentation scheme also applies to the auxiliary information **A**, so as to generate multiple *plaintext segments*, each of which is denoted by $S_{\mathbf{A}}$. Note that some variable-size segmentation schemes [**?**, **?**] identify segment boundaries after the chunks whose contents match a specific pattern and thus address the boundary shift problem faced by the fixed-size segmentation scheme. However, we cannot use these variable-size segmentation schemes [**?**, **?**] in the attack. The reason is that the original contents of chunks in ciphertext segments are protected by symmetric encryption, and we cannot ensure that the boundaries for both ciphertext and plaintext segments match the same pattern. This leads to an incompatibility between ciphertext and plaintext segments, and degrades the amount of segment-level data inferred by the clustering-based attack (see the late part of this section).

We propose to infer ciphertext-plaintext pairs through similar segments. Let $S_{\mathbf{M}}$ be the original plaintext segment of a ciphertext segment $S_{\mathbf{C}}$ (i.e., each plaintext in $S_{\mathbf{M}}$ corresponds to some ciphertext in $S_{\mathbf{C}}$ and vice versa). According to Broder's theorem, if $S_{\mathbf{M}}$ shares the same minimum chunk hash, say h, with some plaintext segment $S_{\mathbf{A}}$, then $S_{\mathbf{M}}$ and $S_{\mathbf{A}}$ tend to have a large fraction of identical plaintexts. This implies that the ciphertexts in $S_{\mathbf{C}}$ are likely to be mapped from the plaintexts in $S_{\mathbf{A}}$. In other words, we first classify all plaintext segments of **A** by their minimum chunk hashes, and obtain multiple *plaintext clusters*. Each plaintext cluster, denoted by $\mathcal{C}_{\mathbf{A}} = \{S_{\mathbf{A}}\}$, corresponds to a unique minimum chunk hash shared by its included segments. We also group the ciphertext segments, whose original plaintext segments have the same minimum chunk hash h (i.e., similar), into an identical *ciphertext cluster*, denoted by $\mathcal{C}_{\mathbf{C}} = \{S_{\mathbf{C}}\}$. Then, we infer the original

data of $\mathcal{C}_{\mathbf{C}}$ from some $\mathcal{C}_{\mathbf{A}}$ that corresponds to h.

We propose a *clustering* scheme to group similar ciphertext segments. A naïve approach is to classify segments by their minimum chunk hashes, but it does not work on ciphertext segments, whose original contents are protected by symmetric encryption. We address the classification issue based on Broder's theorem. Our insight is that if two ciphertext segments have a large fraction of identical ciphertexts, then they correspond to the same minimum chunk hash with a high probability. The reason is that deterministic encryption preserves the cardinalities of union and intersection of plaintext segments, based on which we can use Broder's theorem to learn the equivalence of their minimum chunk hashes.

In this work, we define the *clustering distance* $d(S_{\mathbf{C}}, S'_{\mathbf{C}})$ of any two ciphertext segments $S_{\mathbf{C}}$ and $S'_{\mathbf{C}}$ by one minus the fraction of their identical ciphertexts:

$$d(S_{\mathbf{C}}, S'_{\mathbf{C}}) = 1 - \frac{|S_{\mathbf{C}} \cap S'_{\mathbf{C}}|}{|S_{\mathbf{C}} \cup S'_{\mathbf{C}}|}.$$

Note that identical ciphertexts may repeat in $S_{\mathbf{C}}$ or $S'_{\mathbf{C}}$, and $|S_{\mathbf{C}} \cap S'_{\mathbf{C}}|$ and $|S_{\mathbf{C}} \cup S'_{\mathbf{C}}|$ return the number of *unique* ciphertexts in their intersection and union, respectively. Clearly, the smaller $d(S_{\mathbf{C}}, S'_{\mathbf{C}})$ is, the more likely are $S_{\mathbf{C}}$ and $S'_{\mathbf{C}}$ to correspond to the same minimum chunk hash. Then, we adopt the *agglomerative hierarchical clustering (AHC)* [**?**] to aggregate similar ciphertext segments based on their distance information. Specifically, we start with eash ciphertext segment in its own singleton cluster, and iteratively combine the two closest clusters based on the maximum distance of their ciphertext segments. We configure a parameter k, and stop the iterated combination when the maximum distance of ciphertext segments in the two cloest clusters is greater than k.

For each aggregated ciphertext cluster $\mathcal{C}_{\mathbf{C}}$, we relate it to some plaintext cluster $\mathcal{C}_{\mathbf{A}}$ with frequency analysis, while taking frequency distribution into account. This is based on the observation that identical ciphertexts (resp. plaintexts) may repeat in the same or different ciphertext (resp. plaintext) segments and identical ciphertext (resp. plaintext) segments may also repeat in the same ciphertext (resp. plaintext) cluster. We propose to examine the frequency distribution of the logical ciphertexts or plaintexts in each cluster, and perceive that the frequency distributions for similar clusters (i.e., correspond to the same minimum chunk hash) are also likely to be similar.

We proceed the frequency analysis scheme as follows. First, we sort available ciphertext and plaintext clusters by the total number of logical ciphertexts and plaintexts they

include, respectively. Then, we count an associative array **F** that stores the frequency of each unique ciphertext or plaintext in corresponding cluster. Based on **F**, we compute the probability that a ciphertext C exists in a ciphertext cluster $\mathcal{C}_{\mathbf{C}}$ and further the entropy of $\mathcal{C}_{\mathbf{C}}$:

$$\Pr[C \in \mathcal{C}_{\mathbf{C}}] = \frac{\mathbf{F}[\mathcal{C}_{\mathbf{C}}][C]}{\sum_{C' \in \mathcal{C}_{\mathbf{C}}} \mathbf{F}[\mathcal{C}_{\mathbf{C}}][C']},$$
$$e(\mathcal{C}_{\mathbf{C}}) = \sum_{C \in \mathcal{C}_{\mathbf{C}}} \log_2 \frac{1}{\Pr[C \in \mathcal{C}_{\mathbf{C}}]},$$

where $\mathbf{F}[\mathcal{C}_{\mathbf{C}}][C]$ stores the frequency of C in $\mathcal{C}_{\mathbf{C}}$. Similarly, we compute the entropy $e(\mathcal{C}_{\mathbf{A}})$ for the plaintext cluster $\mathcal{C}_{\mathbf{A}}$. Like the distribution-based attack (see Section 4.2), we configure the frequency analysis scheme with the parameters (u, r, t), and infer that the ciphertext cluster $\mathcal{C}_{\mathbf{C}}$ is similar to a plaintext cluster $\mathcal{C}_{\mathbf{A}}$, if they satisfy the following requirements:

- The rank of $\mathcal{C}_{\mathbf{C}}$ is not larger than u.
- The rank difference of $\mathcal{C}_{\mathbf{C}}$ and $\mathcal{C}_{\mathbf{A}}$ is not larger than r.
- The difference of $e(\mathcal{C}_{\mathbf{C}})$ and $e(\mathcal{C}_{\mathbf{A}})$ is the smallest and also not larger than t.

Then, for each pair $(\mathcal{C}_{\mathbf{C}}, \mathcal{C}_{\mathbf{A}})$ of similar clusters, we infer ciphertext-plaintext pairs in two levels.

- **Segment-level inference:** If $\mathcal{C}_{\mathbf{C}}$ and $\mathcal{C}_{\mathbf{A}}$ have the same number of logical chunks (i.e., ciphertexts or plaintexts), as well as an identical entropy, this implies that $\mathcal{C}_{\mathbf{C}}$ is exactly mapped from $\mathcal{C}_{\mathbf{A}}$ with a high probability. In this case, we operate attack on the coarse-grained *segment* level. Specifically, we first compute the entropies of each ciphertext segment $S_{\mathbf{C}}$ in $\mathcal{C}_{\mathbf{C}}$ and each plaintext segment $S_{\mathbf{A}}$ in $\mathcal{C}_{\mathbf{A}}$, based on the frequency distributions of their ciphertexts and plaintexts, respectively. We infer that $S_{\mathbf{C}}$ is mapped from $S_{\mathbf{A}}$, if the total numbers of logical chunks, as well as the entropies, of $S_{\mathbf{C}}$ and $S_{\mathbf{A}}$ are identical. Our evaluation shows that the segment-level inference contributes most of the correctly inferred contents in the clustering-based attack (see Section 7.3). It is also possible to further recover each plaintext in these inferred segments with additional adversarial knowledge (e.g., ordering).
- **Chunk-level inference:** If $\mathcal{C}_{\mathbf{C}}$ and $\mathcal{C}_{\mathbf{A}}$ have different numbers of logical chunks or entropies, we apply frequency analysis to operate attack on the fine-grained *chunk* level. Specifically, we sort all unique ciphertexts and plaintexts by their frequencies in $\mathcal{C}_{\mathbf{C}}$ and $\mathcal{C}_{\mathbf{A}}$, respectively, and infer ciphertext-plaintext pairs based on frequency ranks. However, we find the chunk-level inference does not perform well in our experimental

dataset. The possible reason is each cluster includes a large number of logical chunks, which degrades the effectiveness of frequency analysis. Even so, we expect the chunk-level inference can recover more ciphertext-plaintext pairs in practice, especially when the number of logical chunks in some clusters is limited.

5.2.1 Summary:

To summarize, the clustering-based attack exploits similarity, and launches frequency analysis in similar clusters to infer ciphertext-plaintext pairs. In addition to u, r and t, it is configured by the parameter k, which specifies the upper bound distance in combining the closest clusters.

Although possibly affected by the boundary shift of fixed-size segments, we argue that the clustering-based attack is severe against VM disk images. Specifically, a flat VM image file is allocated of a fixed size at the time it is created, and such size cannot be changed during its lifetime. All unused regions in a VM image are initially filled with zero chunks, which can be further re-written for storing additional data in the image. In Section 7.3, we examine the effectiveness of the clustering-based attack against VM images.

第六章 数据集与实际系统调研

表 7-1 Characteristics of experimental datasets.

Dataset	Category	Characteristics in Each Snapshot	
		#Logical (Million)	#Unique (Million)
FSL	User004	1.0-1.3	0.7-0.9
	User007	3.5-5.2	2.3-3.6
	User012	25.0-26.4	8.9-9.7
	User013	1.8-5.7	1.2-4.2
	User015	13.4-20.5	9.0-11.0
	User028	6.0-10.3	3.5-6.8
MS	Win7	61.6-61.8	61.1-61.3
	Serv-03	10.6-10.7	8.4-8.5
	Serv-08	∼6.5	∼3.8
	Vista-B	∼7.6	∼2.0
	Vista-U	∼21.0	∼10.4
VM	User1	2.6	∼0.9
	User2		1.1-1.3
	User3		0.9-1.1
	User4		∼0.9
	User5		0.9-1.0
	User6		0.9-1.1

The notations #Logical and #Unique denote the total number of logical and unique chunks in each experimental snapshot, respectively.

第七章 实验测试与分析

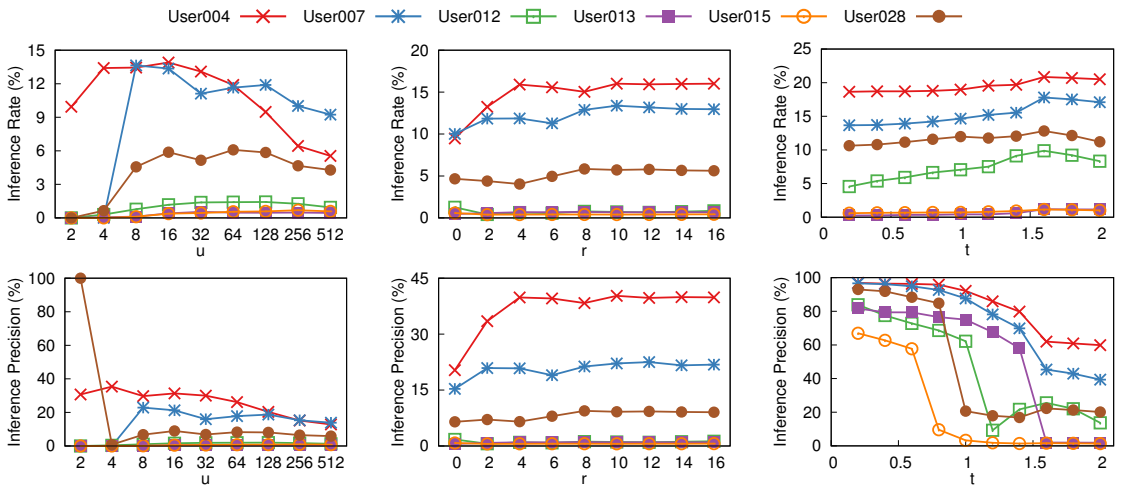
In this section, we present the trace-driven evaluation results to demonstrate the severity of our proposed frequency analysis attacks against encrypted deduplication.

7.1 Methodology

We evaluate our attacks using the following real-world datasets, whose characteristics are summarized in Table 7-1.

- **FSL:** This dataset is collected by the File systems and Storage Lab (FSL) at Stony Brook University [**?**, **?**, 53]. We focus on the *fslhomes* snapshots, each of which includes an ordered list of 48-bit chunk hashes that are produced by variable-size chunking under an average size of 8KB, as well as the corresponding metadata information (e.g., chunk size, file name extension, etc). We pick the snapshots from January 22 to May 21 in 2013, and select six users (i.e., User004, User007, User012, User013, User015, and User028) that have the complete daily snapshots over the whole duration. We collect these snapshots on a weekly basis, and hence form 14 weekly full backups for each user.
- **MS:** This dataset is collected by Microsoft [**?**] and publicized on SNIA [**?**]. The original dataset contains the Windows file system snapshots that span from September 5 to October 31, 2009. Each snapshot is represented by the 40-bit chunk hashes under different average sizes obtained from Rabin fingerprinting [**?**]. We focus on the snapshots that have been installed with the operating systems in the following categories: Windows 7 (Win7), Microsoft Windows Server 2003 (Serv-03), Microsoft Windows Server 2008 (Serv-08), Microsoft Windows Vista Business Edition (Vista-B), and Microsoft Windows Vista Ultimate Edition (Vista-U). In each category, we pick four snapshots, each of which is configured with the average chunk size of 8KB.
- **VM:** This dataset is collected from a university programming course in Spring 2014. The original dataset includes a number of VM image snapshots for the students enrolled in the course, and each snapshot has a fixed size of 10GB and is represented by the ordered list of SHA-1 hashes of 4KB fixed-size chunks. We focus on 6 users (i.e., User1-User6) and extract their snapshots into 13 weekly backups.

Our datasets do not contain actual content, so we mimic the adversarial knowledge based on chunk hashes. Specifically, we use the ordered lists of chunk hashes in some snapshots as the auxiliary information **A** and the ground truth **M**, respectively. To simulate encryption, we apply an additional hash function over each original chunk hash (that represents a plaintext) in **M**, and truncate the result into an agreed number of bits specific to the used dataset. The truncated result mimics a ciphertext in **C**. For each inferred ciphertext-plaintext pair (C, M), we verify its correctness by applying the same simulated encryption on M and comparing the result with C. Note that the clustering-based attack can operate at the segment level, and infer the ciphertext-plaintext segment pair $(S_{\mathbf{C}}, S_{\mathbf{A}})$. In this case, we check $(S_{\mathbf{C}}, S_{\mathbf{A}})$ by examining each ciphertext in $S_{\mathbf{C}}$ is exactly mapped from each plaintext in $S_{\mathbf{A}}$.



(a) Impact of u: $r = 0$; $t \to \infty$

(b) Impact of r: $u = 128$ for User004, User013 and User015, and $u = 256$ for User007, User012 and User028; $t \to \infty$

(c) Impact of t: $u = 128$ for User004, User013 and User015, and $u = 256$ for User007, User012 and User028; $r = 10$

图 7-1 Experiment 1 (Impact of parameters): impact of (u, r, t) in distribution-based attack.

We measure the severity of the attacks by the inference rate and the inference precision (see Section **??**).

7.2 Results of Distribution-based Attack

7.2.1 Experiment 1 (Impact of parameters):

We evaluate the impact of the parameters (u, r, t) in the distribution-based attack. We drive our evaluation using the FSL dataset, and use the 12th weekly backup of each user as the auxiliary information to infer original plaintexts in corresponding 14th weekly backup. We configure $t \to \infty$ and $r = 0$ to evaluate the impact of u (in this case, the distribution-based attack reduces to the locality-based attack [**?**]). Our rationale is to avoid the disturbances by other parameters.

Figure 7-1(a) shows the impact of u, when we vary u from 2 to 512. Regarding inference rate, we have the same observation as the prior work [**?**]. Specifically, the inference rates first increase with u, since the attack can infer more ciphertext-plaintext pairs. After hitting the maximum values (e.g., 13.9% for User004, 13.6% for User007, 1.4% for User012, 0.5% for User013, 0.7% for User015, and 6.1% for User028), they decrease. The reason is that the underlying frequency analysis introduces a large number of false positives that compromise the inferences over corresponding neighbors.

The prior work [**?**] does not report the inference precision about the attack. We observe that the inference precisions for all users are at a fairly low level (e.g., less than 40%), except the case of $u = 2$ for User028 that does not introduce any false positives. On the other hand, the inference rate in the special case is about 0.0001%, meaning that the attack only infers a few ciphertext-plaintext pairs. In addition, as u increases, the inference precisions decrease slightly. For example, when u increases from 2 to 512, the inference precision of User004 drops from 30.7% to 12.8%.

Observation (1) – *A relatively larger u increases the inference rate, yet it decreases the inference precision (i.e., more false positives are introduced).*

Informed by the impact of u, we set u at 128 for User004, User013 and User015, and at 256 for User007, User012 and User028, respectively, in order to evaluate the impact of r and t. Our rationale is to increase the coverage of inferred ciphertext-plaintext pairs, while we use r and t to filter possibly false positives.

We first configure $t \to \infty$ and evaluate the impact of r. Figure 7-1(b) shows the results. We observe that the inference rates of majority users increase with r. For example, when we vary r from 0 to 16, the inference rates grow from 9.5% to 16.0%, from 10.0% to 13.0%, from 0.5% to 0.7%, and from 4.7% to 5.6% for User004, User007, User013, and User028, respectively. The reason is that the distribution-based attack addresses disturbances to frequency ranking, and infer more correct ciphertext-plaintext pairs. On the other hand, the inference rates decrease slightly from 1.3% to 0.8% and from 0.6% to 0.4% for User012 and User015, respectively. The reason is that they examine a large range of plaintexts and may introduce more false positives. In addition, the inference precisions for all users are at a low level (e.g., less than 45%), and have similar tendencies with corresponding inference rates.

Observation (2) – *A larger r provides more opportunities of identifying correct ciphertext-plaintext pairs, yet it also increases the probability of having false positives.*

Then, we fix $r = 10$ and evaluate the impact of t. Figure 7-1(c) shows the results. When t is small (e.g., less than 0.5), we observe that the attack misjudges and filters a significant number of ciphertext-plaintext pairs, even they are correct. This introduces *false negatives* that reduce the inference rate. As t increases, the number of false negatives decreases. When $t = 1.5$, the inference rates hit the maximum values at 21.2%, 18.2%, 10.4%, 1.2%, 1.2%, and 13.5% for User004, User007, User012, User013, User015, and User028, respectively. When t further increases to 2, the corresponding inference rates

drop to 20.5%, 17.1%, 8.3%, 1.2%, 1.0%, and 11.2%, respectively. The reason is that if t is too large, it cannot filter false positives effectively. For the same reason, the inference precisions for all users decrease with t.

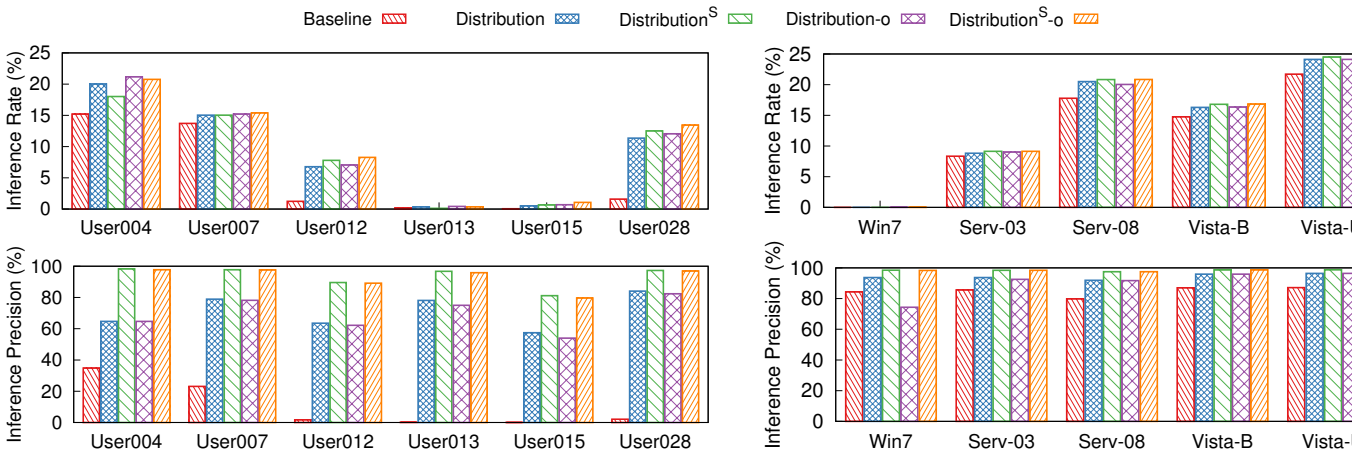
Observation (3) – *A smaller t filters a large fraction of false positives, yet it introduces more false negatives.*

7.2.2 Experiment 2 (Comparison with prior attack):

We compare the severity of the distribution-based attack with that of the locality-based attack [**?**]. In addition to using the FSL dataset like Experiment 1, we include the MS dataset for cross-dataset validation. Specifically, for each MS category, we choose two snapshots at a time, use one to infer the other, and evaluate the average inference rate and precision.

We consider the following attack instances for comparison.

- `Baseline`: We re-implement the locality-based attack based on the parameter configuration suggested in [**?**]. Specifically, it infers 5 most frequent ciphertext-plaintext pairs in the first invocation (i.e., to initialize a set of ciphertext-plaintext pairs for iteration) of frequency analysis, and 30 in each following invocation (i.e., to iteratively infer ciphertext-plaintext pairs from neighbors).
- `Distribution` and $\texttt{Distribution}^{\texttt{S}}$: We consider two attack instances of the distribution-based attack, denoted by $\texttt{Distribution}^{\texttt{S}}$ and `Distribution`, which operate with and without size information, respectively (i.e., the superscript `S` indicates that the attack instance operates with size information). We configure $\texttt{Distribution}^{\texttt{S}}$ and `Distribution` under the same configuration of `Baseline`. In addition, we choose r and t in both $\texttt{Distribution}^{\texttt{S}}$ and `Distribution` in the following way: for the FSL dataset, we fix $r = 10$ for all users, and individually set $t = 1.5$, 1.2, 1, 1, 0.7, and 0.9 for User004, User007, User012, User013, User015, and User028, respectively; for the MS dataset, we also fix $r = 10$ for all categories, and set $t = 2$ for Win7 and Serv-08, and $t = 1.6$ for Vista-U, Serv-03, and Vista-B, respectively. This is informed by our tests for optimal configurations of the datasets.
- `Distribution-o` and $\texttt{Distribution}^{\texttt{S}}\texttt{-o}$: We consider two additional distribution-based attack instances, denoted by `Distribution-o` and $\texttt{Distribution}^{\texttt{S}}\texttt{-o}$, which apply the same configurations for r and t as with `Distribution` and $\texttt{Distribution}^{\texttt{S}}$, and further use larger u to increase the coverage of inferred ciphertext-plaintext pairs. Specifically,



(a) FSL dataset

(b) MS dataset

图 7-2 Experiment 2 (Comparison with prior attack): comparison of attack severity for distribution-based attack and locality-based attack.

we configure u in `Distribution-o` and `Distribution`^{S}`-o` in the following way: for the FSL dataset, we apply the same configuration of u in Experiment 1; for the MS dataset, we set u = 128 for Win7, and u = 30 for Serv-03, Serv-08, Vista-B, and Vista-U.

Figure 7-2(a) shows the comparison results of the FSL dataset. We observe that different instances of the distribution-based attack outperform the locality-based attack in almost all cases. For example, regarding User028, the lowest inference rate of the distribution-based attack is 11.4%, with the precision of 84.1% (due to `Distribution`), while the corresponding inference rate and precision of `Baseline` are only 1.2% and 1.7%, respectively; this implies that the distribution-based attack reduces the number of false postives by 82.4% in this case.

Observation (4) – *The distribution-based attack significantly increases the inference precision, while achieving a higher inference rate than the locality-based attack.*

`Distribution`^{S} and `Distribution`^{S}`-o` have higher inference precisions than `Distribution` and `Distribution-o`, respectively, since they further filter false positives by size information. For example, for User004, `Distribution`^{S} and `Distribution`^{S}`-o` reduce the fraction of false positives in `Distribution` and `Distribution-o` from 35.2% to 1.7% and from 35.3% to 2.2%, respectively. However, in the same case, we observe that the inference rates of `Distribution` and `Distribution-o` are 20.0% and 21.2%, slightly higher than those of `Distribution`^{S} and `Distribution`^{S}`-o` by 2.0% and 0.4%, respectively. The reason is that `Distribution` and `Distribution-o` infer a small number of

correct results from the neighbors of incorrect ciphertext-plaintext pairs. In other words, although (C, M) is an incorrect ciphertext-plaintext pair, the neighbors of C may correspond to those of M with a small probability. Even in this case, all distribution-based attack instances are more severe than the locality-based attack instance. Specifically, the inference rate of `Baseline` is only 15.2%, lower than those of the best and the worst distribution-based attack instances by 6.0% and 2.8%, respectively.

Observation (5) – *Filtering incorrect inference results improves the inference precision, yet it degrades the coverage of inferred ciphertext-plaintext pairs and possibly decreases the inference rate.*

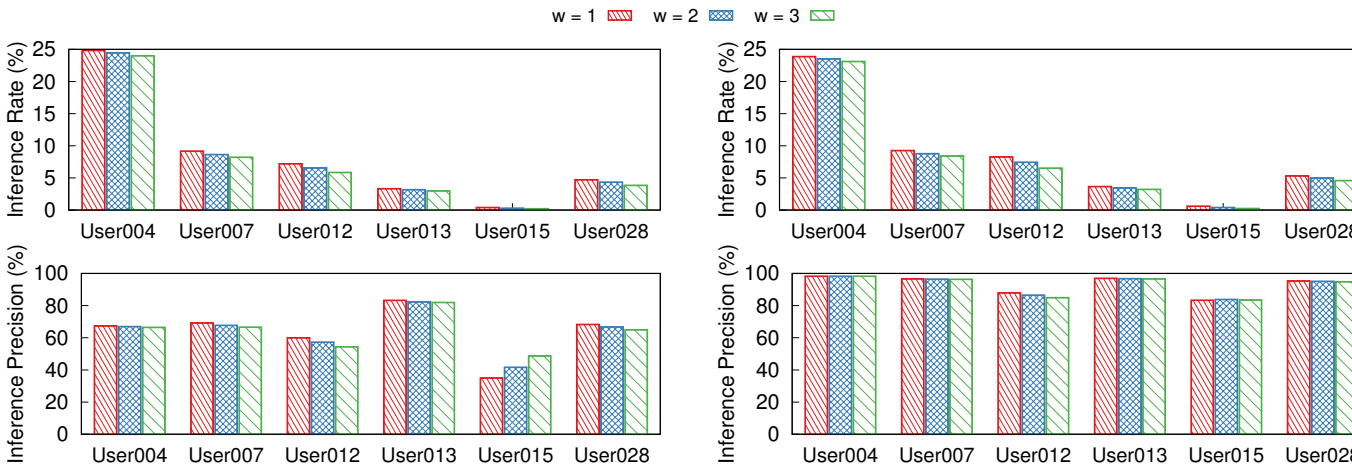
We further observe that although `Distribution-o` and `Distribution`$^{\text{S}}$`-o` build on larger u, their inference rates are just slightly higher than those of `Distribution` and `Distribution`$^{\text{S}}$ by 0.4% and 0.9%, respectively. The reason is that the distribution-based attack iterates inference just through neighbors, and has a bounded coverage of inferred ciphertext-plaintext pairs. The further increase of u only adds a small number of new correct ciphertext-plaintext pairs into results.

Figure 7-2(b) shows the results of the MS dataset. Both locality-based and distribution-based attacks have high inference rates and precisions in most MS categories (except Win7). The possible reason is that MS snapshots are highly correlated (e.g., the variance of the total number of chunks is small, as shown in Table 7-1). We observe that the distribution-based attack still outperforms the locality-based attack. For example, in Vista-U, the inference rates and precisions of all inferences of the distribution-based attack are above 24.1% and 96.4%, while those of `Baseline` are 21.7% and 87.1%, respectively.

Note that both distribution-based and locality-based attacks have low inference rates (e.g., less than 0.01%) in the Win7 category. The reason is that Win7 includes a large fraction (e.g., more than 98.8%, as shown in Table 7-1) of unique chunks, which cannot be correctly inferred by the frequency analysis attacks.

7.2.3 Experiment 3 (Attack effectiveness):

We consider a long-term backup scenario and examine the effectiveness of the distribution-based attack with the FSL dataset. Specifically, we choose the ith FSL weekly backup of each user as the auxiliary information to infer original plaintexts in the corresponding $(i+w)$th FSL weekly backup. Clearly, the smaller w is, the higher correlation between the auxiliary information and the target backup will be. We configure the two distribution-



(a) Distribution-based attack without size information

(b) Distribution-based attack with size information

图 7-3 Experiment 3 (Attack effectiveness): severity of distribution-based attack in FSL dataset.

based attack instances `Distribution-o` and `Distribution`^{S}`-o` like Experiment 2, and evaluate their inference rates and inference precisions that are averaged for all available i for each user.

Figure 7-3(a) shows the results. The distribution-based attack has varying inference rate and precision across users. For example, in the favorable case like User004, it achieves the highest inference rates of 24.8%, 24.5%, and 24.0% with the precisions of 67.3%, 67.0%, and 66.5% for w = 1, 2, and 3, respectively; in the non-favorable case like User015, the inference rate of the distribution-based attack is only around 0.3%. The possible reason is that the backup data from User015 has low chunk locality.

In addition, we observe that the correlation (i.e., w) of the auxiliary information has low impact on the effectiveness of the distribution-based attack. For example, when w increases from 1 to 3, it only leads to limited degradations on inference rate (e.g., less than 1.4%) and precision (e.g., less than 5.6%). The reason is that the distribution-based attack addresses disturbances to frequency ranking and preserves attack effectiveness.

Observation (6) – *The distribution-based attack can limit the degradation of attack effectiveness in the presence of loosely correlated auxiliary information.*

Figure 7-3(b) shows the results of `Distribution`^{S}`-o`. We observe that it has similar inference rate with `Distribution-o`, while achieving much higher precision. For example, the average inference precision of all users are 93.1%, 92.8%, and 92.4% for w = 1, 2, and 3, respectively.

7.3 Results of Clustering-based Attack

7.3.1 Experiment 4 (Impact of parameter):

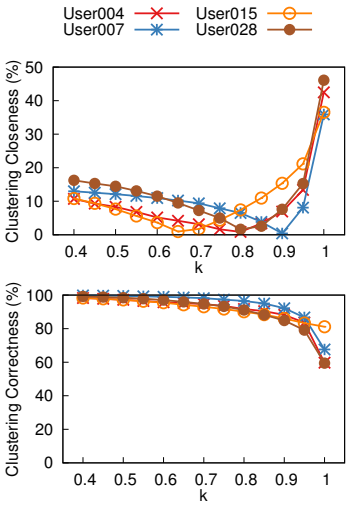
We first evaluate the impact of the parameter k, which defines the upper bound distance in combining the closest clusters. We use both the FSL and the VM datasets to study how k affects the underlying clustering scheme in the attack. Specifically, we apply segmentation on the last backup of each considered FSL and VM user, respectively, and generate the segments that have a fixed size of 4MB.

The clustering scheme aims to aggregate similar ciphertext segments into the same cluster, without compromising the confidentiality of chunks in each ciphertext segment. To quantify its effect, we compare the results of clustering with those of a *real* classification approach, which directly classifies segments by their minimum chunk hash. Suppose we generate m real classes of segments by classification, and $\widetilde{m}$ clusters by the clustering scheme, respectively. We consider *clustering closeness*, evaluated by $\frac{\mathsf{abs}(m-\widetilde{m})}{m}$ (where $\mathsf{abs}(m-\widetilde{m})$ returns the absolute value of $m-\widetilde{m}$), which quantifies how the number of clusters approximates that of real classes. In addition, let $\hat{m}$ be the number of clusters, in which all segments are similar (i.e., have the same minimum chunk hash). We also consider *clustering correctness*, evaluated by $\frac{\hat{m}}{\widetilde{m}}$, which quantifies how precisely the clustering scheme groups similar segments.

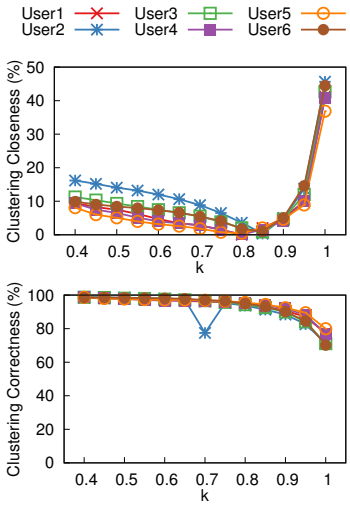
Figure 7(a) shows the results of the FSL dataset, where we consider four FSL users (e.g., User004, User007, User015 and User028) for saving evaluation time. The clustering closeness first increases with k, since the number (i.e., $\widetilde{m}$) of clusters decreases and approximates m. When k increases further, the number of clusters drops away from m, and leads to the increase of clustering closeness. In addition, we observe that the clustering correctness gradually decreases with k, because some of non-similar segments (i.e., their minimum chunk hashes are different) are aggregated into the same cluster. Both results suggest that we can configure an appropriate k to balance the closeness and correctness of clustering. For example, when we set $k = 0.65$ for User015, the corresponding closeness and correctness are 1.0% and 94.2%, respectively. This implies that the results of the clustering scheme highly approximates those of real classification.

Observation (7) – *By configuring an appropriate k for clustering, we approximate the results of classifying segments, without the knowledge of minimum chunk hash in each segment.*

Figure 7(b) shows the results of the VM dataset. We observe that the clustering



(a) FSL dataset



(a) VM dataset

Fig. 7 Experiment 4 (Impact of parameter): impact of k in the clustering scheme; for clustering closeness, the smaller the better; for clustering correctness, the larger the better.

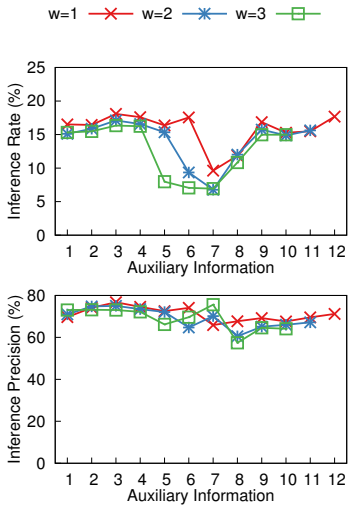


Fig. 8 Experiment 5 (Attack effectiveness): severity of clustering-based attack in VM dataset.

closeness and correctness of all VM users have similar tendencies with those of FSL users. When we configure $k = 0.8$, the average clustering closeness of all VM users is only 3.0%, while the corresponding clustering correctness is as high as 93.1%.

7.3.2 Experiment 5 (Attack effectiveness):

We now study the effectiveness of the clustering-based attack. Due to the boundary shift of fixed-size segment, it has low effectiveness (about 1% inference rate in our test) against the FSL dataset. Thus, we use the VM dataset to examine its severity. To configure the attack, we set $k = 0.8$ for clustering, and $(u, r, t) = (5000, 100, 0.5)$ for relating ciphertext clusters to corresponding plaintext clusters.

In our micro-benchmarking, we find that the chunk-level inference in the clustering-based attack only infers thousands of chunks correctly, which contributes to a negligible inference rate (e.g., less than 0.01%). Thus, we focus on the segment-level inference, which presents the bottom line of severity in the clustering-based attack. To be consistent with the chunk-level measurements, we count inference rate and precision based on the unique chunks in each correctly inferred segment.

We use the same evaluation methodology of Experiment 3, and report the results in Figure 8. Specifically, the x-axis describes the index i (where $1 \leq i \leq 12$) of the VM backup that is used as the auxiliary information for attack, while the y-axis presents the

表 7-2 Experiment 6 (Security implications)

(a) Distribution-based attacks

Type	Extension Name	Range of File Size	Raw Inference Rate	
			Distribution-o	Distribution^S-o
Office	doc(x), ppt(x), xls(x)	10KB-1MB	4.9%	5.3%
Picture	jpg, png	10-100KB	7.7%	6.7%
Source	c, h, cpp, java, py	10-20KB	17.1%	15.0%
Database	db, po	20-700KB	2.6%	2.4%
Disk	vmdk, img	200MB-1GB	15.8%	16.7%

(b) Clustering-based attack

Users	Raw Inference Rate
1	13.2%
2	22.4%
3	25.7%
4	28.4%
5	18.5%
6	30.8%

average inference rate or precision of all VM users against the $(i+w)$th backup (where w = 1, 2, and 3). We observe that both the inference rate and the inference precision fluctuate significantly. For example, when using the 3rd backup as the auxiliary information, the attack achieves the highest inference rates at 18.1%, 17.1% and 16.3%, with the precisions of 76.8%, 75.0% and 73.0% for w = 1, 2, and 3, respectively. On the other hand, when using the 7th backup as the auxiliary information, the corresponding inference rates and precisions drop down to 9.6% and 65.9%, 6.8% and 71.2%, and 6.9% and 75.6%, respectively. The reason is that the VM users have heavy updates after the 7th week, and this reduces the correlation of adjacent backups. On average, for w = 1, 2 and 3, the clustering-based attack infers 15.8%, 14.0%, and 12.6% ciphertext-plaintext pairs, with the precisions of 71.1%, 69.1%, and 68.9%, respectively.

7.4 Results of Security Implications

We thus far have examined the severity of inference attacks by quantifying the correctly inferred ciphertext-plaintext pairs. However, it remains an open issue that what are the security implications informed by these results and how the frequency analysis attacks bring actual damage. In the following experiment, we evaluate the security implications of our attacks based on the *raw inference rate*, defined as the percentage of raw data content affected by correctly inferred chunks.

7.4.1 Experiment 6 (Security implications):

We first consider the distribution-based attack, and evaluate its raw inference rate against *different types* of files. We drive the evaluation using the FSL dataset, since only the FSL dataset includes file metadata (that includes the extension names of files) in plaintext. Specifically, we focus on five types of files that have specific extension names (see Table 7-2(a)): office files (*Office*), picture files (*Picture*), programming source files (*Source*), database files (*Database*), and disk image files (*Disk*). These files occupy more than 60% of raw content of FSL snapshots.

We apply the methodology of Experiment 2, and evaluate the raw inference rates of Distribution^{S}-o and Distribution-o. Table 7-2(a) shows the results. Both attack instances have high raw inference rates against *Disk* (e.g., 15.8% for Distribution-o and 16.7% for Distribution^{S}-o), because each disk file includes a large number of rarely updated chunks that form high locality within the same file. Interestingly, we observe that *Source*, although each file is of a small size, also incurs high raw inference rates by the distribution-based attacks (e.g., 17.1% for Distribution-o and 15.0% for Distribution^{S}-o). The reason is that programming source files are often stored together in file system (e.g., the source files that belong to the same project locate in an identical directory) and form a large stretch of correlated chunks, which present high locality across files. For small and scattered files (e.g., *Office*, *Picture*, and *Database*), the distribution-based attacks have relative low raw inference rates.

Observation (8) – *The severity of the distribution-based attack depends on the update frequencies, sizes, and spatiality of target files.*

We examine the security implication of the clustering-based attack using the VM dataset. Specifically, we use the 11th backup of each VM user to infer original content in corresponding 13th VM backup. Since the VM dataset does not contain any metadata,

we count the raw inference rate based on the whole data content in each VM snapshot. Note that we filter all zero chunks in the count of raw inference rate, because they occupy a large fraction in VM disk images [**?**].

We use the same configuration of Experiment 5, and evaluate raw inference rate based on segment-level inference. Table 7-2(b) shows the results for different users. We observe that the clustering-based attack achieves high severity against the VM dataset. For example, it infers up to 30.8% raw content of User6's VM backup. On average, the raw inference rate of all users is as high as 23.2%.

Observation (9) – *The clustering-based attack threatens the confidentiality of VM disk images.*

第八章 应对频率分析攻击的对策讨论

In this section, we discuss the advantages and disadvantages of possible countermeasures against the leakage channels exploited in this paper. Note that the countermeasures against size leakage are not enough for preventing our attacks, in which the size information is just optional.

8.1 Against frequency leakage:

MinHash encryption [**?**, **?**] encrypts each plaintext with a key derived from the minimum chunk hash over a set of its adjacent chunks, and possibly maps identical plaintexts into different ciphertexts. The rationale for defense is that MinHash encryption changes frequencies, and disturbs frequency ranking. Note that MinHash encryption also prevents the frequency analysis attacks in this paper, because we target deterministic encryption (e.g., MLE [**?**]), while MinHash encryption is non-deterministic and changes the frequency distribution of chunks. The disadvantage is that MinHash encryption degrades the storage saving by deduplication, since it breaks the one-to-one mapping between plaintexts and ciphertexts. In addition, MinHash encryption is not an active countermeasure, since its randomness essentially depends on the minimum chunk hashes in the target workloads.

A recent work [**?**] suggests intentionally adding duplicate chunks to prevent traffic analysis against client-side deduplication. The approach [**?**] can be applied to address the frequency analysis attacks, since it changes the frequencies of chunks. Compared with MinHash encryption, the countermeasure [**?**] does not degrade storage efficiency, since only duplicate chunks are added. On the other hand, it requires the priori knowledge that if particular chunks are duplicate, and only suits the client-side deduplication, which possibly introduces additional leakage channels (see Section **??**).

Several extended MLE instantiations build on strong cryptographic primitives to defend against the frequency leakage of encrypted deduplication, such as randomized encryption that supports equality testing [**?**], hybrid encryption [**?**], and interactions with fully homomorphic encryption [**?**]. They provide provable security, yet how they can be implemented and deployed in practice remains unexplored.

8.2 Against order leakage:

A simple countermeasure is to disturb the deduplication processing sequence of chunks. For example, we can operate an additional order-scrambling process on the stream of plaintexts before encryption, so as to hide the actual logical order of each plaintext. This prevents the distribution-based attack, because the adversary cannot identify neighboring information correctly. On the other hand, it is not always effective against the clustering-based attack. If scrambling operates on just a small basis (e.g., the orders of plaintexts in each segment are scrambled), the clustering-based attack still works. The disadvantage of the order-scrambling countermeasure is that it breaks chunk locality and leads to performance drop in large-scale deduplication [**?**, **?**, **?**].

8.3 Against size leakage:

As suggested by prior work [**?**], we can pad each plaintext with additional data to obfuscate the actual size of this plaintext. However, we note that the padding scheme is not straightforward, since it requires to add identical redundancies to the same plaintexts; otherwise the storage system cannot detect duplicates for deduplication. One possible solution is to build on the paradigm of MLE [**?**, **?**]. We compute the cryptographic hash of each plaintext as a seed, and use it to generate variable-size pseudorandom data to be padded with the plaintext. Like MLE, this solution comes at the expense of server-aided approach [**?**] to defend against brute-force attack (see Section **??**).

Instead of variable-size chunking, we can use fixed-size chunking to generate equal-size chunks for deduplication. Since all chunks have the same size, the adversary cannot exploit size information to differentiate them. Although fixed-size chunking suffers from boundary shift, it achieves almost the same storage saving of deduplication as variable-size chunking in VM disk images [**?**]. Thus, we suggest applying fixed-size chunking in its favorable workloads to prevent size leakage.

第九章 相关工作

MLE instantiations: Recall from Section 第二章 that MLE [**?**] formalizes the cryptographic foundation of encrypted deduplication. The first published MLE instantiation is convergent encryption (CE) [**?**], which uses the cryptographic hash of a plaintext and its corresponding ciphertext as the MLE key and the tag, respectively. Other CE variants include: hash convergent encryption (HCE) [**?**], which derives the tag from the plaintext while still using the hash of the plaintext as the MLE key; random convergent encryption (RCE) [**?**], which encrypts a plaintext with a fresh random key to form a non-deterministic ciphertext, protects the random key by the MLE key derived from the hash of the plaintext, and attaches a deterministic tag derived from the plaintext for duplicate checks; and convergent dispersal (CD) [**?**], which extends CE to secret sharing by using the cryptographic hash of a plaintext as the random seed of a secret sharing algorithm. Since all the above instantiations derive MLE keys and/or tags from the plaintexts *only*, they are vulnerable to the offline brute-force attack [**?**] if the plaintext is *predictable* (i.e., the number of all possible plaintexts is limited) , as an adversary can exhaustively derive the MLE keys and tags from all possible plaintexts and check if any plaintext is encrypted to a target ciphertext (in CE, HCE, and CD) or mapped to a target tag (in RCE). The brute-force attack has been demonstrated to learn file information [**?**].

To protect against the offline brute-force attack, DupLESS [**?**] implements server-aided MLE by managing MLE keys in a standalone key server, which ensures that each MLE key cannot be derived from a message offline. DupLESS employs two mechanisms to achieve robust MLE key management: (i) oblivious key generation, in which a client always obtains a deterministic MLE key for a message from the key server without revealing the message content to the key server, and (ii) rate limiting, which limits the key generation requests from the client and prevents the online brute-force attack. Other studies extend server-aided MLE to address various aspects, such as reliable key management [**?**], transparent pricing [**?**], peer-to-peer key management [**?**], and rekeying [**?**]. However, server-aided MLE still builds on deterministic encryption. To our knowledge, existing MLE instantiations (based on either CE or server-aided MLE) are all vulnerable to the inference attacks studied in this paper.

9.1 Attacks against (encrypted) deduplication:

In addition to the offline brute-force attack, previous studies consider various attacks against deduplication storage, and such attacks generally apply to encrypted deduplication as well. For example, the side-channel attack [**?**, **?**] enables adversaries to exploit the deduplication pattern to infer the content of uploaded files from target users or gain unauthorized access in client-side deduplication; it is shown that the side-channel attack (and other related attacks) was successfully launched against Dropbox in 2010 [**?**]. The duplicate-faking attack [**?**] compromises message integrity via inconsistent tags. Ritzdorf *et al.* [**?**] exploit the leakage of the chunk size to infer the existence of files. The locality-based attack [**?**] exploits frequency analysis to infer ciphertext-plaintext pairs. Our work follows the line of work on inference attacks [**?**, **?**], yet provides a more in-depth study of inference attacks against encrypted deduplication via various types of leakage.

9.2 Defense mechanisms:

Section **??** discusses the countermeasures agasint the frequency, order and size leakage of encrypted deduplication. Additional defense mechanisms are designed to protect against other types of attacks. As mentioned above, server-aided MLE [**?**] can defend against the offline brute-force attack. Server-side deduplication [**?**, **?**, **?**] and proof-of-ownership [**?**, **?**, **?**] can defend against the side-channel attack. Server-side tag generation [**?**, **?**] and guarded decryption [**?**] can defend against the duplicate-checking attack.

9.3 Inference attacks:

Several inference attacks have been proposed against encrypted databases [**?**, **?**, **?**, **?**, **?**, **?**] and keyword search [**?**, **?**, **?**, **?**, **?**]. They all exploit the deterministic encryption nature to identify different types of leakage. Our work differs from them by specifically focusing on frequency analysis against encrypted deduplication.

第十章 结论

Encrypted deduplication applies deterministic encryption, and leaks the frequencies of plaintexts. This paper revisits the security vulnerability due to frequency analysis, and demonstrates that encrypted deduplication is even more vulnerable towards inference attacks. We propose two new frequency analysis attacks, both of which achieve high inference rate and high inference precision, while under different assumptions of adversarial knowledge. We empirically evaluate our attacks with three real-world datasets, present a variety of new observations about their natures, and further analyze how they bring actual damages. We also discuss the advantages and disadvantages of possible countermeasures to advise practitioners for securely implementing and deploying encrypted deduplication storage systems.

We pose three directions for future work. First, we consider a complete old backup as the auxiliary information, and do not study how to launch attack if the adversary only has partial knowledge about the old backup. Possibly, we can still apply the frequency analysis attacks to extract characteristics from the available partial backup, and infer ciphertext-plaintext pairs by comparing the characteristics with those in the target backup. One direction is can we design advanced inference attacks, which perform better than directly applying our attacks in this partial knowledge case?

Second, we adjust the parameters of the frequency analysis attacks based on their effectiveness, which can only be learnt after the attacks have happened. We do not study how to derive the optimal parameters from auxiliary information beforehand. Future work may do better.

Third, we do not implement attack prototypes against real-world encrypted deduplication storage systems. Another direction is to deploy our attack design and report the vulnerability of encrypted deduplication in practice.

参考文献

[1] Gartner. Gartner forecasts 59 percent mobile data growth worldwide in 2015[EB/OL]. http://www.gartner.com/newsroom/id/3098617, Dec 12, 2018

[2] 敖莉, 舒继武, 李明强. 重复数据删除技术 [J]. 软件学报, 2010, 21(5): 916-929

[3] J. McKnight, T. Asaro, B. Babineau. Digital archiving: End-user survey and market forecast 2006–2010[J]. The Enterprise Strategy Group, 2006,

[4] 付印金, 肖侬, 刘芳. 重复数据删除关键技术研究进展 [J]. 计算机研究与发展, 2012, 49(1): 12-20

[5] EMC. Emc data domain[EB/OL]. http://www.emc.com/en-us/data-protection/data-domain.htm, Dec 12, 2018

[6] EMC. Avamar deduplication backup software and system[EB/OL]. http://www.emc.com/data-protection/avamar.htm, Dec 12, 2018

[7] veritas. Netbackup appliances[EB/OL]. https://www.veritas.com/product/backup-and-recovery/netbackup-appliances.html, Dec 12, 2018

[8] CommVault. Commvault solutions for data protection backup and recovery[EB/OL]. http://www.commvault.com/solutions/by-function/data-protection-backup-and-recovery, Dec 12, 2018

[9] D. Harnik, B. Pinkas, A. Shulman-Peleg. Side channels in cloud services: Deduplication in cloud storage[J]. IEEE Security & Privacy, 2010, 6): 40-47

[10] M. Bellare, S. Keelveedhi, T. Ristenpart. Message-locked encryption and secure deduplication[C]. Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2013, 296-312

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

[12] M. Naveed, S. Kamara, C. V. Wright. Inference attacks on property-preserving encrypted databases[C]. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015, 644-655

[13] MEGA. Mega[EB/OL]. https://mega.nz, Dec 12, 2018

[14] ElephantDrive. Elephantdrive[EB/OL]. https://www.elephantdrive.com, Dec 12, 2018

[15] Cryptosphere. Cryptosphere[EB/OL]. https://cryptosphere.io, Dec 12, 2018

[16] Freenet. Freenet[EB/OL]. https://freenetproject.org, Dec 12, 2018

[17] GNU. Gnu's framework for secure peer-to-peer networking[EB/OL]. https://gnunet.org, Dec 12, 2018

[18] Tahoe-LAFS. Tahoe-lafs[EB/OL]. https://tahoe-lafs.org/trac/tahoe-lafs., Dec 12, 2018

[19] M. Li, C. Qin, J. Li, et al. Cdstore: Toward reliable, secure, and cost-efficient cloud storage via convergent dispersal[J]. IEEE Internet Computing, 2016, 3): 45-53

[20] S. Keelveedhi, M. Bellare, T. Ristenpart. Dupless: server-aided encryption for deduplicated storage[C]. USENIX Security 13, 2013, 179-194

[21] M. Li, C. Qin, P. P. Lee. Cdstore: Toward reliable, secure, and cost-efficient cloud storage via convergent dispersal.[C]. USENIX Annual Technical Conference, 2015, 111-124

[22] Y. Duan. Distributed key generation for encrypted deduplication: Achieving the strongest privacy[C]. Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security, 2014, 57-68

[23] F. Armknecht, J.-M. Bohli, G. O. Karame, et al. Transparent data deduplication in the cloud[C]. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015, 886-900

[24] J. Liu, N. Asokan, B. Pinkas. Secure deduplication of encrypted data without additional independent servers[C]. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015, 874-885

[25] Y. Zhou, D. Feng, W. Xia, et al. Secdep: A user-aware efficient fine-grained secure deduplication scheme with multi-level key management[C]. Mass Storage Systems and Technologies (MSST), 2015 31st Symposium on, 2015, 1-14

[26] J. Li, J. Li, D. Xie, et al. Secure auditing and deduplicating data in cloud[J]. IEEE Transactions on Computers, 2016, 65(8): 2386-2396

[27] J. Li, C. Qin, P. P. Lee, et al. Rekeying for encrypted deduplication storage[C]. Dependable Systems and Networks (DSN), 2016 46th Annual IEEE/IFIP International Conference on, 2016, 618-629

[28] C. Qin, J. Li, P. P. Lee. The design and implementation of a rekeying-aware encrypted deduplication storage system[J]. ACM Transactions on Storage (TOS), 2017, 13(1): 9

[29] M. Abadi, D. Boneh, I. Mironov, et al. Message-locked encryption for lock-dependent messages[M]. Springer, 2013, 374-391

[30] J. Stanek, A. Sorniotti, E. Androulaki, et al. A secure data deduplication scheme for cloud storage[C]. International Conference on Financial Cryptography and Data Security, 2014, 99-118

[31] M. Bellare, S. Keelveedhi. Interactive message-locked encryption and secure deduplication[C]. IACR International Workshop on Public Key Cryptography, 2015, 516-538

[32] J. Katz, A. J. Menezes, P. C. Van Oorschot, et al. Handbook of applied cryptography[M]. CRC press, 1996

[33] R. Kumar, J. Novak, B. Pang, et al. On anonymizing query logs via token-based hashing[C]. Proceedings of the 16th international conference on World Wide Web, 2007, 629-638

[34] D. Cash, P. Grubbs, J. Perry, et al. Leakage-abuse attacks against searchable encryption[C]. Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, 2015, 668-679

[35] P. Grubbs, R. McPherson, M. Naveed, et al. Breaking web applications built on top of encrypted data[C]. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, 1353-1364

[36] M. S. Islam, M. Kuzu, M. Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation.[C]. Ndss, 2012, 12

[37] D. Pouliot, C. V. Wright. The shadow nemesis: Inference attacks on efficiently deployable, efficiently searchable encryption[C]. Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, 2016, 1341-1352

[38] Y. Zhang, J. Katz, C. Papamanthou. All your queries are belong to us: The power of file-injection attacks on searchable encryption.[C]. USENIX Security Symposium, 2016, 707-720

[39] V. Bindschaedler, P. Grubbs, D. Cash, et al. The tao of inference in privacy-protected databases[J]. Proceedings of the VLDB Endowment, 2018, 11(11): 1715-1728

[40] F. B. Durak, T. M. DuBuisson, D. Cash. What else is revealed by order-revealing encryption?[C]. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, 1155-1166

[41] P. Grubbs, K. Sekniqi, V. Bindschaedler, et al. Leakage-abuse attacks against order-revealing encryption[C]. Security and Privacy (SP), 2017 IEEE Symposium on, 2017, 655-672

[42] G. Kellaris, G. Kollios, K. Nissim, et al. Generic attacks on secure outsourced databases[C]. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, 1329-1340

[43] M.-S. Lacharité, B. Minaud, K. G. Paterson. Improved reconstruction attacks on encrypted data using range query leakage[C]. 2018 IEEE Symposium on Security and Privacy (SP), 2018, 297-314

[44] J. Li, C. Qin, P. P. Lee, et al. Information leakage in encrypted deduplication via frequency analysis[C]. Dependable Systems and Networks (DSN), 2017 47th Annual IEEE/IFIP International Conference on, 2017, 1-12

[45] B. Zhu, K. Li, R. H. Patterson. Avoiding the disk bottleneck in the data domain deduplication file system.[C]. Fast, 2008, 1-14

[46] M. Lillibridge, K. Eshghi, D. Bhagwat, et al. Sparse indexing: Large scale, inline deduplication using sampling and locality.[C]. Fast, 2009, 111-123

[47] W. Xia, H. Jiang, D. Feng, et al. Silo: A similarity-locality based near-exact deduplication scheme with low ram overhead and high throughput.[C]. USENIX annual technical conference, 2011, 26-30

[48] S. Halevi, D. Harnik, B. Pinkas, et al. Proofs of ownership in remote storage systems[C]. Proceedings of the 18th ACM conference on Computer and communications security, 2011, 491-500

[49] M. Mulazzani, S. Schrittwieser, M. Leithner, et al. Dark clouds on the horizon: Using cloud storage as attack vector and online slack space.[C]. USENIX security symposium, 2011, 65-76

[50] H. Ritzdorf, G. Karame, C. Soriente, et al. On information leakage in deduplicated storage systems[C]. Proceedings of the 2016 ACM on Cloud Computing Security Workshop, 2016, 61-72

[51] D. Bhagwat, K. Eshghi, D. D. Long, et al. Extreme binning: Scalable, parallel deduplication for chunk-based file backup[C]. Modeling, Analysis & Simulation of Computer and Telecommunication Systems, 2009. MASCOTS'09. IEEE International Symposium on, 2009, 1-9

[52] Z. Sun, G. Kuenning, S. Mandal, et al. A long-term user-centric analysis of deduplication patterns[C]. Mass Storage Systems and Technologies (MSST), 2016 32nd Symposium on, 2016, 1-7

[53] FSL. FSL traces and snapshots public archive[EB/OL]. http://tracer.filesystems.org/, Dec 12, 2018

[54] D. T. Meyer, W. J. Bolosky. A study of practical deduplication[J]. ACM Transactions on Storage (TOS), 2012, 7(4): 14

[55] OpenDedup. SDFS: Opensource dedup to cloud and local storage[EB/OL]. http://opendedup.org/odd/documentation/, Dec 12, 2018

[56] M. Fu. Destor: a platform for data deduplication evaluation[EB/OL]. https://github.com/fomy/destor, Dec 12, 2018

[57] J. Black. Compare-by-hash: A reasoned analysis.[C]. USENIX Annual Technical Conference, General Track, 2006, 85-90

[58] M. O. Rabin. Fingerprinting by random polynomials[J]. Technical report, 1981,

[59] K. Jin, E. L. Miller. The effectiveness of deduplication on virtual machine disk images[C]. Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference, 2009, 7

[60] A. Adya, W. J. Bolosky, M. Castro, et al. Farsite: Federated, available, and reliable storage for an incompletely trusted environment[J]. ACM SIGOPS Operating Systems Review, 2002, 36(SI): 1-14

[61] L. P. Cox, C. D. Murray, B. D. Noble. Pastiche: Making backup cheap and easy[J]. ACM SIGOPS Operating Systems Review, 2002, 36(SI): 285-298

[62] M. W. Storer, K. Greenan, D. D. Long, et al. Secure data deduplication[C]. Proceedings of the 4th ACM international workshop on Storage security and survivability, 2008, 1-10

[63] P. Anderson, L. Zhang. Fast and secure laptop backups with encrypted de-duplication.[C]. LISA, 2010,

[64] Z. Wilcox-O'Hearn, B. Warner. Tahoe: the least-authority filesystem[C]. Proceedings of the 4th ACM international workshop on Storage security and survivability, 2008, 21-26

附　录