Revisiting Frequency Analysis against Encrypted Deduplication via Statistical Distribution

Jingwei Li[†], Guoli Wei[†], Jiacheng Liang[†], Yanjing Ren[†], Patrick P. C. Lee[‡], and Xiaosong Zhang[†,*]
[†]*University of Electronic Science and Technology of China* [‡]*The Chinese University of Hong Kong*
[*]*Blockchain Research Lab of UESTC, Chengdu Jiaozi Financial Holding Group Company Ltd*

***Abstract*—Encrypted deduplication addresses both security and storage efficiency in large-scale storage systems: it ensures that each plaintext is encrypted to a ciphertext by a symmetric key derived from the content of the plaintext, so as to allow deduplication on the ciphertexts derived from duplicate plaintexts. However, the deterministic nature of encrypted deduplication leaks the frequencies of plaintexts, thereby allowing adversaries to launch frequency analysis against encrypted deduplication and infer the ciphertext-plaintext pairs in storage. In this paper, we revisit the security vulnerability of encrypted deduplication due to frequency analysis, and show that encrypted deduplication can be even more vulnerable to the sophisticated frequency analysis attack that exploits the underlying storage workload characteristics. We propose the distribution-based attack, which builds on a statistical approach to model the relative frequency distributions of plaintexts and ciphertexts, and improves the inference precision (i.e., have high confidence on the correctness of inferred ciphertext-plaintext pairs) of the previous attack. We evaluate the new attack against real-world storage workloads and provide insights into its actual damage.**

I. Introduction

Modern large-scale storage systems reduce storage space by removing content redundancy via *deduplication*, which stores only the data copies with unique content among already stored data copies. Field studies show that deduplication reduces substantial fractions of storage space under real-world storage workloads, such as backups [35], file system snapshots [27], and virtual disk images [18]. Cloud storage services (e.g., Dropbox and Google Drive) also use deduplication to save storage costs [16]. To ensure data confidentiality, clients should encrypt their own data before uploading the data to the cloud. It is critical that the encryption approach preserves the original content redundancy pattern, so that duplicate data copies can still be deduplicated even after encryption.

We study *encrypted deduplication*, which combines encryption and deduplication in a way that each plaintext (data copy) is symmetrically encrypted and decrypted by a secret key that is derived from the content of the plaintext itself; should the plaintexts be duplicates, their resulting ciphertexts are also duplicates and can be deduplicated. Encrypted deduplication can be achieved via the formal cryptographic primitive called *message-locked encryption (MLE)* [8], whose instantiations have been extensively studied and realized in the literature (see Section VI). MLE builds on *deterministic encryption*, which deterministically maps a plaintext to the same ciphertext to preserve the identical content pattern after encryption.

Corresponding author: Patrick P. C. Lee (pclee@cse.cuhk.edu.hk)

However, the deterministic nature of encrypted deduplication causes information leakage, such that the adversaries can analyze the ciphertexts and infer the corresponding original plaintexts. Specifically, an adversary can perform *frequency analysis* [3] on both the ciphertexts that are observed and the plaintexts that are known as priori. Then the adversary can infer the ciphertext-plaintext pairs, in which both the ciphertext and plaintext in each pair have correlated frequency patterns. Such inference attacks have been shown effective against database records [29] and searchable keywords [11], [17] that are protected by deterministic encryption. Recently, frequency analysis is also shown effective against encrypted deduplication [21].

In this paper, we revisit the information leakage in encrypted deduplication. We show that encrypted deduplication can actually be *even more vulnerable* to the sophisticated frequency analysis attack that exploits the workload characteristics of real-world storage patterns. In addition to inferring a significant fraction of ciphertext-plaintext pairs, such a sophisticated attack can reduce the false positives of inference (i.e., an inferred plaintext is correctly mapped to a target ciphertext with a high probability).

We propose a new frequency analysis attack, namely the *distribution-based attack*, which targets backup workloads [35] in encrypted deduplication. We build on the *locality* property [25], [36], which states that the ordering of data is likely to be preserved across various backups. We extend locality from the statistics perspective, and propose a new workload characteristics measurement, namely the *relative frequency distribution*, which characterizes the likelihood of the co-occurrence of data in the logical order. We find that the relative frequency distributions of identical data are stable in different backups. Thus, the distribution-based attack aims to ensure high confidence on the (inferred) ciphertext-plaintext pairs, by examining the relative frequency distributions of the ciphertext and plaintext in each pair to filter falsely inferred ciphertext-plaintext pairs.

We evaluate the distribution-based attack with real-world datasets. On average, it ensures that 83.6% (higher than the previous attack [21] that achieves only 19.2%) of inferred ciphertext-plaintext pairs are correct, while the correctly inferred ones take 48.7% (also higher than the previous attack [21] that achieves 12.6%) of all the ciphertext-plaintext pairs in total (including both inferred and non-inferred ones). In addition, we find that the distribution-based attack preserves

TABLE I
MAJOR NOTATIONS USED IN THIS PAPER.

Notation	Description
Defined in Section II	
C (M)	Unique ciphertext (plaintext)
$\hat{C}^{(i)}$ ($\hat{M}^{(i)}$)	The i-th logical ciphertext (plaintext)
$\mathbf{C}$ ($\mathbf{M}$)	Sequence of logical ciphertexts (plaintexts)
$\mathbf{A}$	Auxiliary information represented by sequence of logical plaintexts
$\mathbf{L_C}$ ($\mathbf{R_C}$)	Associative array that maps each unique ciphertext to co-occurrence frequencies with its left (right) neighbors in $\mathbf{C}$
$\mathbf{L_A}$ ($\mathbf{R_A}$)	Associative array that maps each unique plaintext to co-occurrence frequencies with its left (right) neighbors in $\mathbf{A}$
u	Number of ciphertext-plaintext pairs returned by the first iteration of frequency analysis in locality-based attack
Defined in Section III	
X	Random variable that describes left neighboring chunk
Y	Random variable that describes right neighboring chunk
$\mathbf{L_M}[M]$	Set of left neighbors of M in $\mathbf{M}$
$\mathbf{R_M}[M]$	Set of right neighbors of M in $\mathbf{M}$
$\mathbf{L_M}[M][M']$	Co-occurrence frequency of M and its left neighbor M'
$\mathbf{R_M}[M][M']$	Co-occurrence frequency of M and its right neighbor M'
C_i (M_i)	i-th frequent ciphertext (plaintext)
r	Range of frequency ranks to be examined for each ciphertext
t	Euclidean distance threshold

high attack severity, even when the priori knowledge is loosely correlated with the target contents. This highlights the attack severity, and should be taken into consideration when developing countermeasures.

II. Background and Problem

In this section, we present the basics of encrypted deduplication. We elaborate the threat model and review previous frequency analysis attacks and their weaknesses. Table I summarizes the major notations used in this paper.

A. Basics

We focus on *chunk-based deduplication* that operates at the granularity of small-size data units called *chunks*. Specifically, a storage system partitions a file (e.g., backup) into a list of fixed-size or variable-size (e.g., via Rabin fingerprinting [31]) *logical chunks* (a.k.a. logical plaintexts), denoted by $\mathbf{M} = \langle \hat{M}^{(1)}, \hat{M}^{(2)}, \ldots \rangle$. Each logical chunk is uniquely identified by the cryptographic hash of its content called a *tag* (a.k.a. fingerprint). Two logical chunks are said to be *identical* if they have the same tag, while the likelihood that distinct logical chunks have the same tag is assumed to be negligible [10]. Since identical logical chunks may appear multiple times in $\mathbf{M}$, the storage system stores each *unique chunk* (a.k.a. unique plaintext) M that has the content matching one or many logical chunks, and refers the corresponding logical chunks to the stored unique chunk via small-size references.

Encrypted deduplication maps each logical plaintext $\hat{M}^{(i)}$ into the corresponding ciphertext $\hat{C}^{(i)}$ via symmetric encryption, so as to address chunk confidentiality in an outsourcing environment (e.g., cloud storage, see Figure 1). Specifically, multiple clients encrypt logical chunks in the client sides, and outsource the storage of the ciphertexts in a cloud. The cloud storage system performs deduplication by tags (see above) and only stores the physical chunks that have unique contents.

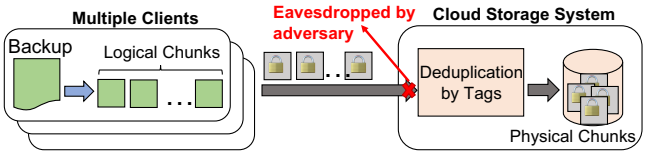


Fig. 1. Architecture of encrypted deduplication: an adversary can eavesdrop the sequence of logical ciphertexts before deduplication.

To realize encrypted deduplication, traditional symmetric encryption requires that multiple clients encrypt their plaintexts by their (distinct) secret keys, thereby converting identical plaintexts into distinct ciphertexts that can no longer be deduplicated. *Message-locked encryption (MLE)* [8] derives a symmetric key (called the *MLE key*) from the content of each logical plaintext, and encrypts the plaintext using the MLE key to form a *logical ciphertext* (e.g., an encrypted logical chunk). MLE ensures that identical plaintexts are always encrypted to identical ciphertexts, and hence the storage system can derive the tag from each ciphertext to perform deduplication.

Note that MLE builds on *deterministic encryption* to ensure that the ciphertexts (or the tags) are deterministically derived from the plaintexts, so as to preserve deduplication effectiveness as opposed to traditional symmetric encryption. While deterministic encryption provides confidentiality guarantees, we show how an adversary can exploit the deterministic nature to infer the original plaintext from a given ciphertext in the context of encrypted deduplication.

B. Threat Model

We target periodic backups that are created as the complete copies of primary data (e.g., file system snapshots) on a daily or weekly basis. We consider a passive adversary that monitors the stream of logical ciphertexts $\mathbf{C} = \langle \hat{C}^{(1)}, \hat{C}^{(2)}, \ldots \rangle$ being written to the cloud storage system (see Figure 1). For example, the adversary compromises the deduplication process, identifies the characteristics of logical ciphertexts, and aims to learn the information about the corresponding plaintexts. Specifically, suppose that $\mathbf{M}$ is the sequence of logical plaintexts corresponding to $\mathbf{C}$. We focus on two leakage channels that can be exploited by the adversary:

- **Frequency.** Due to the deterministic nature of MLE-based encrypted deduplication, the frequency (i.e., the number of duplicate copies) of each unique ciphertext in $\mathbf{C}$ reflects the frequency of its corresponding plaintext in $\mathbf{M}$.
- **Order.** Many storage systems (e.g., [36]) apply deduplication to the chunks in the same order as they appear in the original file. Thus, the order of the logical ciphertexts in $\mathbf{C}$ reflects the order of the logical plaintexts in $\mathbf{M}$.

We assume that the adversary knows some *auxiliary information* that presents the ground truth about the characteristics correlated with $\mathbf{M}$. The availability of the auxiliary information is necessary for any inference attack (e.g., [9], [14], [20], [29]), and this paper considers the auxiliary information as an ordered list of previously known plaintexts (e.g., old

backups), denoted by $\mathbf{A}$. Note that the attack severity (e.g., high inference rate and inference precision, see below) depends on the correlation between $\mathbf{A}$ (i.e., the previously known plaintexts) and $\mathbf{M}$ (i.e., the plaintexts that are to be inferred). Our focus is not to address how an adversary can obtain auxiliary information (e.g., due to careless data release [1]). Instead, we focus on how the available auxiliary information, when combined with the leakage channels, brings information leakage to encrypted deduplication. Our evaluation (Exp#3) also shows that even when $\mathbf{A}$ is loosely correlated with $\mathbf{M}$, the information leakage is still significant.

Given the stream of ciphertexts in $\mathbf{C}$ and the available auxiliary information in $\mathbf{A}$, the adversary infers the ciphertext-plaintext pairs, denoted by $\{(C, M)\}$, with two goals:

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

Our threat model does not rely on other adversarial capabilities that can be prevented by existing approaches. For example, 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 typically do not apply deduplication to the metadata and it can be protected by traditional symmetric encryption. Also, we assume that the adversary does not have any *active* capability (e.g., tampering stored chunks), which can be prevented by remote integrity checking [5], [19].

C. Previous Attacks

Frequency analysis is a classical attack methodology to infer ciphertext-plaintext pairs against deterministic encryption. It [3] sorts the unique ciphertexts in $\mathbf{C}$ and the unique plaintexts in $\mathbf{A}$ by frequency. It relates each unique ciphertext C in $\mathbf{C}$ with the unique plaintext M in $\mathbf{A}$, such that both M and C have the same frequency rank. However, the classical frequency analysis brings negligible severity to encrypted deduplication, since many updates may occur from $\mathbf{A}$ (i.e., a previous backup) to $\mathbf{C}$ (i.e., the recent backup) and disturb the frequency ranks of identical chunks [21].

The *locality-based attack* [21] augments frequency analysis with *locality*, which is commonly found in backup workloads [25], [36]. Locality states that the neighboring chunks tend to co-occur in the *same order* across different versions of backups prior to deduplication. The rationale is that the updates to each backup are often clustered in small regions of chunks, while the remaining large stretch of chunks stay intact and preserve the same order across different versions of backups.

Based on locality, the locality-based attack exploits the order leakage to discover the neighboring information of ciphertexts and plaintexts. Specifically, for a given unique ciphertext C, the attack first identifies the set of all corresponding duplicate copies $\{\hat{C}^{(i)}\}$. For each $\hat{C}^{(i)}$, it examines the left and right neighbors of $\hat{C}^{(i)}$ (i.e., $\hat{C}^{(i-1)}$ and $\hat{C}^{(i+1)}$, respectively), and 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 from each unique ciphertext C and any of its left and right neighbor (e.g., $\hat{C}^{(i-1)}$ and $\hat{C}^{(i+1)}$) to the corresponding *co-occurrence frequency*. Similarly, the attack also constructs the associative arrays $\mathbf{L_A}$ and $\mathbf{R_A}$ based on the order information of $\mathbf{A}$.

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 $\mathbf{C}$ and $\mathbf{A}$. The inferred results are likely to be *correct* (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 (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.

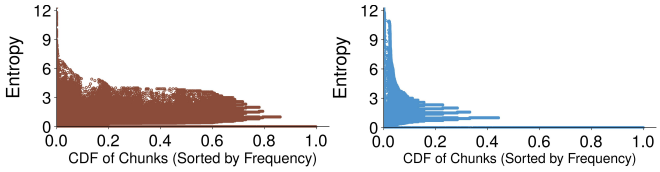
Motivation. The locality-based attack has a major weakness that it introduces a high number of false positives (i.e., incorrectly inferred ciphertext-plaintext pairs) in the inference results. Since the main idea of frequency analysis is to map ciphertexts to the plaintexts with the same frequency ranks, any disturbance to frequency ranking (e.g., the updates across backups) can lead to incorrectly inferred ciphertext-plaintext pairs and in turn compromise the inference of their neighbors. Specifically, although the locality-based attack can infer many ciphertext-plaintext pairs that collectively cover a significant fraction of all correct ciphertext-plaintext pairs (i.e., high inference rate) [21], *the adversary has low confidence to tell whether each inferred ciphertext-plaintext pair is correct or in fact a false positive (i.e., low inference precision).* In this paper, we extend locality via statistical distribution and design a more severe frequency analysis attack against encrypted deduplication, with high inference rate and high inference precision.

III. Distribution-based Attack

In this section, we propose to explore the *relative frequency distributions* of each chunk, and then present the *distribution-based attack*, which examines the relative frequency distributions to improve inference precision.

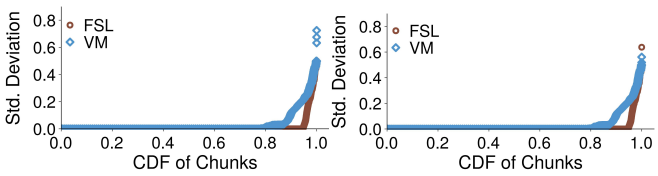
A. Relative Frequency Distributions

Definitions. Recall that a unique plaintext M may repeat in a stream of plaintexts $\mathbf{M}$, and hence has multiple left and right neighbors. We define the relative frequency distributions of



(a) FSL dataset (b) VM dataset

Fig. 2. Entropy (i.e., $e(\mathbf{L_M}[M])$) of each chunk based on its left neighbors in backup workloads FSL and VM: all chunks are sorted by frequency and each point corresponds to the entropy of a unique chunk.



(a) Left neighbors (b) Right neighbors

Fig. 3. Standard deviations of each chunk's entropies in different backups.

M based on two random variables, X and Y, which describe the co-occurrences of M with its left and right neighbors, respectively. Specifically, the event "$X = M'$" denotes the case that M' is the left neighbor of M, while the event "$Y = M'$" denotes the case that M' is the right neighbor of M. Suppose that $\mathbf{L_M}[M]$ and $\mathbf{R_M}[M]$ store the left and right neighbors of M, respectively. We define the *probability dense functions* of the relative frequency distributions of M as follows:

$$\Pr[X = M'] = \frac{\mathbf{L_M}[M][M']}{\sum_{M'' \in \mathbf{L_M}[M]} \mathbf{L_M}[M][M'']}, \quad (1)$$

$$\Pr[Y = M'] = \frac{\mathbf{R_M}[M][M']}{\sum_{M'' \in \mathbf{R_M}[M]} \mathbf{R_M}[M][M'']}, \quad (2)$$

where $\mathbf{L_M}[M][M']$ and $\mathbf{R_M}[M][M']$ store the number of co-occurrences, in which M' is the left and right neighbors of M in $\mathbf{M}$, respectively. We treat the relative frequency distributions as generalized notions of locality. Specifically, they extend locality to characterize the co-occurrence likelihood of each chunk with its neighbors.

Trace-driven analysis. We now analyze the relative frequency distributions of the chunks in the real-world backup datasets, namely *FSL* and *VM* (see Section IV-A for the dataset details). To compare the relative frequency distributions of different chunks, we characterize the relative frequency distributions by *entropies*, which quantify the randomness of the variables X and Y that describe the co-occurrences of neighboring chunks. Specifically, corresponding to the Equations (1) and (2), we define the entropies of the relative frequency distributions of M as:

$$e(\mathbf{L_M}[M]) = \sum_{M' \in \mathbf{L_M}[M]} \Pr[X = M'] \log_2 \frac{1}{\Pr[X = M']}, \quad (3)$$

$$e(\mathbf{R_M}[M]) = \sum_{M' \in \mathbf{R_M}[M]} \Pr[Y = M'] \log_2 \frac{1}{\Pr[Y = M']}, \quad (4)$$

where $\mathbf{L_M}[M]$ and $\mathbf{R_M}[M]$ store all left and right neighbors of M, respectively, and $\Pr[X = M']$ and $\Pr[Y = M']$ are the probability dense functions of the relative frequency distributions of M.

We first compare the entropies of different chunks. We merge all FSL (e.g., 16 FSL backups) and VM (e.g., 13 VM backups) backups, respectively, and compute the entropies of each unique chunk based on its relative frequency distributions (i.e., via Equations (3) and (4)). Figure 2 shows the entropies based on the left neighbors of the chunks (that are sorted by frequency), and the results for right neighbors are similar and we omit them here. We observe that the entropies of *different* frequent chunks are varying, significantly. For example, the entropy values of the top-100 frequent chunks vary from 0.002 to 12.3 in FSL and from 0.005 to 12.2 in VM. On the other hand, since the in-frequent chunks have a limited number of neighbors, their entropies only have a few possible values. In the extreme case, the entropy of the chunks that appear only once (taking 14.0% of all unique chunks in FSL and 55.8% of all unique chunks in VM) is always zero.

Then, we focus on the chunks that exist in all backups, and compare the entropies of identical chunks in different backups. Specifically, we compute each unique chunk's entropies (based on the corresponding left and right neighbors via Equations (3) and (4), respectively) in every backup, and derive the *standard deviations* among its entropies in all backups. Clearly, a small standard deviation implies that the probabilities that a chunk co-occurs with other chunks are stable across different backups. Figure 3 shows the distributions of the standard deviations of all chunks. The results based on the left and right neighbors are similar. Specifically, a large fraction (e.g., for FSL, 94.7% based on both left and right neighbors; for VM, 78.3% and 79.4% based on left and right neighbors, respectively) of chunks only have a low standard deviation (e.g., zero).

In summary, our analysis finds that *different frequent chunks generally have varying relative frequency distributions (Figure 2), while the relative frequency distributions of identical chunks are stable across distinct backups (Figure 3).* The new findings inspire us to use the relative frequency distribution as a necessary condition to filter unreasonable inference results, so as to improve the inference precision of frequency analysis.

B. Attack Description

The distribution-based attack extends the locality-based attack [22] to remove false positives. In addition to leveraging locality as in the locality-based attack, it measures the relative frequency distributions of each unique ciphertext C in $\mathbf{C}$, as well as those of each unique plaintext M in $\mathbf{A}$. It filters the (possibly incorrect) inferred ciphertext-plaintext pairs (C, M) if the relative frequency distributions of C and M are significantly distinct. The rationale is that C and M are now likely to correspond to different chunks.

Figure 4 presents the workflow of the distribution-based attack. It first sorts the unique ciphertexts and plaintexts by their frequencies in $\mathbf{C}$ and $\mathbf{A}$, respectively. As in the

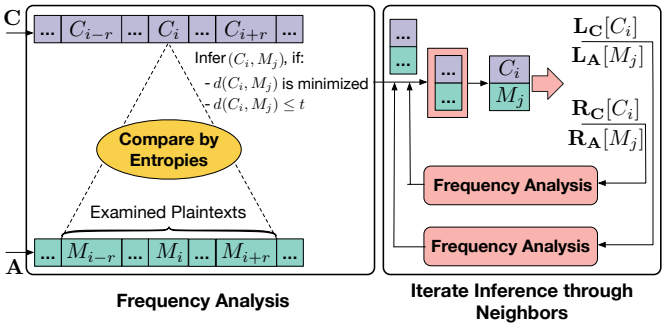


Fig. 4. Workflow of distribution-based attack.

locality-based attack, it configures the parameter u (e.g., 64 by default), so the underlying frequency analysis returns at most u ciphertext-plaintext pairs. For each unique ciphertext C_i of rank i ($1 \le i \le u$), it examines a number of unique plaintexts $M_{i-r}, \ldots, M_i, \ldots, M_{i+r}$ ranking from $i-r$ to $i+r$, where r is a configurable parameter (e.g., 12 by default) that describes the range of plaintexts to be examined for each unique ciphertext. Specifically, the parameter r provides robustness against the disturbance of frequency rankings of the ciphertexts and plaintexts (a major weakness in the locality-based attack as stated in Section II-C), since the attack now matches each unique ciphertext with the plaintext in the r-range (see below).

For C_i ($1 \le i \le u$) and each of the corresponding M_j ($i-r \le j \le i+r$), the distribution-based attack compares their relative frequency distributions by entropies. Specifically, it computes the entropies of C_i in $\mathbf{C}$: $e(\mathbf{L_C}[C_i])$ and $e(\mathbf{R_C}[C_i])$, as well as those of each M_j in $\mathbf{A}$: $e(\mathbf{L_A}[M_j])$ and $e(\mathbf{R_A}[M_j])$ (via Equations (1)-(4)). It compares the entropies of C_i and M_j by the *Euclidean distance*, denoted by $d(C_i, M_j)$:

$$d(C_i, M_j) = \{[e(\mathbf{L_C}[C_i]) - e(\mathbf{L_A}[M_j])]^2 + [e(\mathbf{R_C}[C_i]) - e(\mathbf{R_A}[M_j])]^2\}^{1/2}. \quad (5)$$

C_i and M_j have similar relative frequency distributions if the Euclidean distance $d(C_i, M_j)$ of their entropies is small. Thus, the attack identifies (C_i, M_j) as an inferred ciphertext-plaintext pair if it satisfies the following requirements:

- **R1:** $d(C_i, M_j)$ is the *smallest* for all $i - r \le j \le i + r$.
- **R2:** $d(C_i, M_j)$ is at most a pre-defined parameter t (e.g., 1 by default).

Through R1, the attack infers the ciphertext-plaintext pair (C_i, M_j), when their relative frequency distributions are the most similar. Note that the original plaintext of C_i may fall outside the examined plaintexts $M_{i-r}, \ldots, M_{i+r}$ in some extreme cases. Then, R1 is still satisfied by some M_j ($i-r \le j \le i+r$), and we expect to filter the incorrectly inferred ciphertext-plaintext pairs by R2.

The distribution-based attack follows the iteration paradigm in the locality-based attack (see Section II-C) to increase the coverage of inferred ciphertext-plaintext pairs. Specifically, for each inferred (C_i, M_j), it applies the above frequency analysis approach to infer more ciphertext-plaintext pairs through the neighbors of C_i and M_j, and iterates on those newly inferred pairs until no more ciphertext-plaintext pairs can be inferred.

Summary. The distribution-based attack provides a generalized notion of locality by considering the relative frequency distributions during frequency analysis. It is configured by three parameters (i) u, which specifies the maximum number of ciphertext-plaintext pairs returned by the underlying frequency analysis, (ii) r, which specifies the range of rank disturbance to be addressed, and (iii) t, which specifies the Euclidean distance threshold to filter possibly incorrect inference results. The previous locality-based attack [21] can be viewed as a special case of the distribution-based attack under the parameter configuration of $r = 0$ (i.e., without addressing the disturbance to frequency ranking) and $t \to \infty$ (i.e., without filtering any incorrect inference results).

IV. Attack Evaluation

We evaluate the distribution-based attack based on backup workloads. We also study its security damage on different files.

A. Setup

We drive our evaluation based on two real-world datasets.

- **FSL.** This dataset [34] is collected based on the snapshots of nine students' home directories in a shared network file system. Each snapshot includes many files and is represented as the metadata of each file, as well as an ordered list of the 48-bit hashes of corresponding chunks that are obtained from variable-size chunking. We consider the snapshots, whose chunks have an average chunk size of 8 KiB. We pick all nine students' snapshots from January 22 to May 21 in 2013, aggregate them on a weekly basis and obtain 16 weekly-full backups. In summary, each FSL backup includes about 28-32 million unique chunks, and the deduplication ratio of all backups is 22.4×.
- **VM.** This dataset is collected based on the virtual machine (VM) image snapshots for the students enrolled in a university programming course. Each snapshot is represented as an ordered list of SHA-1 hashes of 4 KiB fixed-size chunks. Like the previous work [21], we extract 13 VM backups in a weekly basis, and remove the zero chunks that are known to dominate the storage of VM images [18]. Specifically, each VM backup includes about 3-13 million unique chunks, and the overall deduplication ratio is 69.2×.

Note that our datasets do not contain actual content, so we mimic the adversarial knowledge based on chunk hashes. Specifically, we select the original snapshots as either the auxiliary information $\mathbf{A}$ or the ground truth $\mathbf{M}$. To simulate deterministic MLE, we apply an additional hash function over each original chunk hash (representing a plaintext) in $\mathbf{M}$ to form a ciphertext in $\mathbf{C}$. For each inferred ciphertext-plaintext pair (C, M), we verify its correctness by applying the same simulated MLE on M and comparing the result with C. By default, we configure u = 64, r = 12 and t = 1 for the distribution-based attack, and evaluate the attack severity by the inference rate and inference precision (defined in Section II-B).

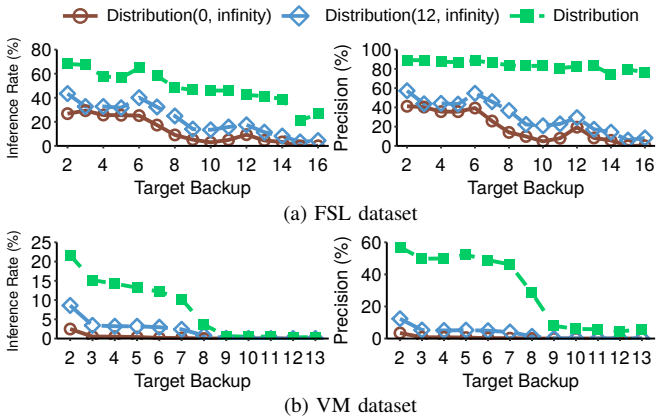


Fig. 5. Varying target backups (Exp#1).

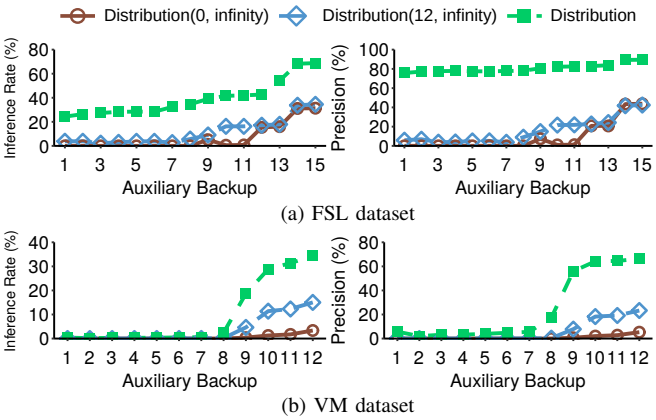


Fig. 6. Varying auxiliary backups (Exp#2).

B. Attack Severity

We evaluate the attack severity of the distribution-based attack, in order to justify the necessities of the parameters r (that is used to address disturbance to frequency ranking) and t (that is used to filter unreasonable inference results). Specifically, we fix u at 64 and consider three instances: (i) `Distribution`$(0, \infty)$, which does not use r or t; (ii) `Distribution`$(12, \infty)$, which fixes r at 12 and does not use t; and (iii) `Distribution`, which applies our default configuration. Note that the first instance `Distribution`$(0, \infty)$ is equivalent to the previous locality-based attack [21].

Exp#1: Varying target backups (Figure 5). We first fix the auxiliary information as the first backup, and vary the target backup for attack. Figure 5 shows the inference rate and precision for FSL and VM datasets, respectively. Obviously, the attack effectiveness of all instances is generally higher when the auxiliary and target backups are closer, since the auxiliary backup now is more correlated with the target backup.

In addition, we observe that `Distribution` is more severe than the other two instances. In FSL, it achieves the average inference rate of 48.7% with the precision of 83.6%, significantly higher than those of `Distribution`$(0, \infty)$ (whose inference rate and precision are 12.6% and 19.2%, respectively) and `Distribution`$(12, \infty)$ (whose inference rate and precision are 21.7% and 31.1%, respectively). The reason is in two-fold. On the one hand, it finds more correct pairs, in which the ciphertext and plaintext are under different ranks (i.e., by addressing ranking disturbances). On the other hand, it ensures high confidence on inferred pairs (i.e., by filtering false positives), and in turn helps iteratively infer more correct pairs from corresponding neighbors.

We have similar observations for VM (Figure 5(b)). When we vary the target backup beyond the eighth VM backup, the attack effectiveness decreases significantly, since a huge update occurs between the eighth and the ninth backups. Even so, `Distribution` is more severe than `Distribution`$(0, \infty)$ and `Distribution`$(12, \infty)$ for all target VM backups.

Exp#2: Varying auxiliary backups (Figure 6). Next, we fix the target backup as the last backup, and vary the auxiliary backup for attack. Figure 6 shows the results for FSL and VM datasets, respectively. `Distribution` again outperforms `Distribution`$(0, \infty)$ and `Distribution`$(12, \infty)$. For example, the average inference rate and precision for `Distribution` are 39.1% and 80.7% in FSL, respectively, while those are 6.7% and 9.2% for `Distribution`$(0, \infty)$, as well as 11.6% and 15.6% for `Distribution`$(12, \infty)$.

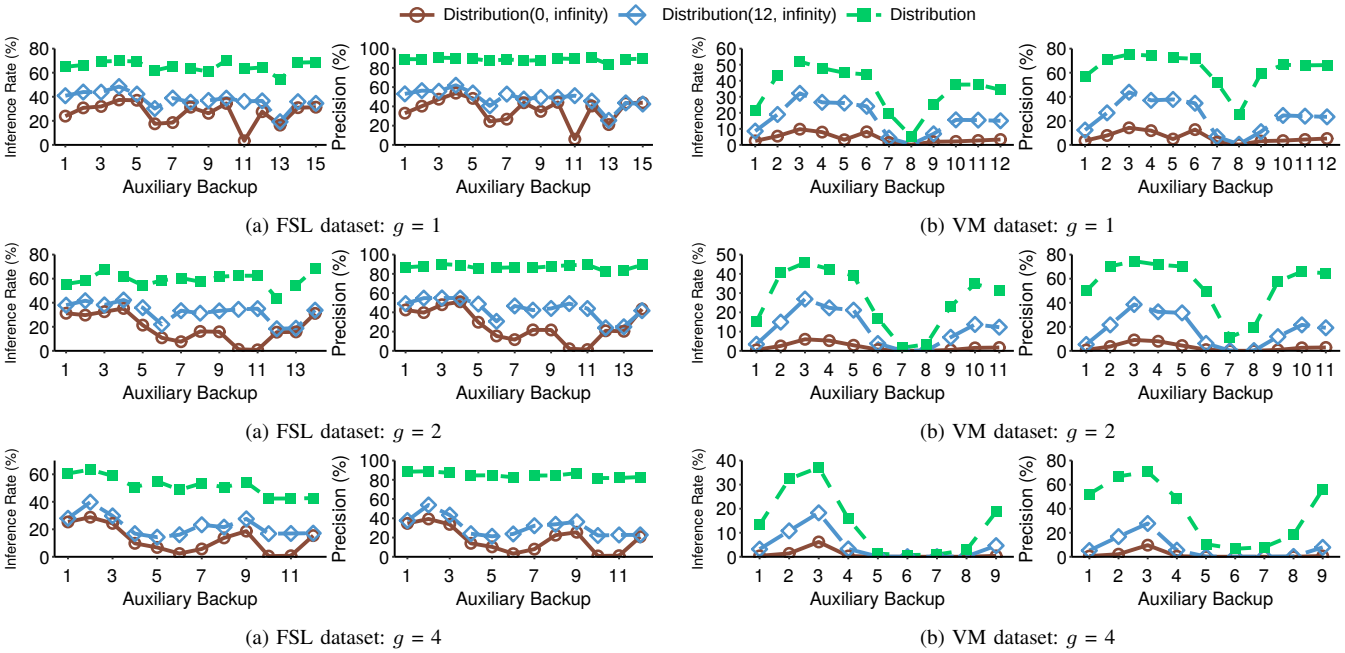
In VM, due to the huge update after the eighth backup, we only have significant attack effectiveness starting from using the ninth backup as auxiliary information. Specifically, when we use the ninth backup as the auxiliary information, the inference rate of `Distribution` is 18.7%, about 49.4× of `Distribution`$(0, \infty)$ and 4.0× of `Distribution`$(12, \infty)$. Also, the corresponding inference precision of `Distribution` is 56.0%, about 89.2× of `Distribution`$(0, \infty)$ and 7.1× of `Distribution`$(12, \infty)$.

Exp#3: Attacks using a sliding window (Figure 7). We launch the considered attacks based on a sliding window approach. We choose the i-th backup as the auxiliary information, and infer the original plaintexts in the $(i+g)$-th backup, while we vary i and g in our evaluation (the smaller g is, the more correlated the auxiliary and the target backups are).

Figure 7 shows the results for g = 1, 2, and 4. Intuitively, a large sliding window size should decrease attack severity, significantly. However, in FSL, when we increase g from 1 to 4, the average inference rate of `Distribution` (relatively) slightly decreases from 65.4% to 51.9% (i.e., by the percentage of 20.6%), while `Distribution` always preserves high inference precision (e.g., 87.0-89.1%). On the other hand, the average inference rate of `Distribution`$(0, \infty)$ drops from 26.6% to 12.7% (i.e., by the percentage of 52.3%) and that of `Distribution`$(12, \infty)$ drops from 37.5% to 22.3% (i.e., by the percentage of 40.1%). Also, both `Distribution`$(0, \infty)$ and `Distribution`$(12, \infty)$ have low inference precision (e.g., less than 32%) when g = 4.

Summary. We summarize our observations about the severity of the distribution-based attack as follows.

- On average, it improves the inference rate and precision of the previous attack [21] (i.e., `Distribution`$(0, \infty)$) by 3.9× and 4.4×, respectively.

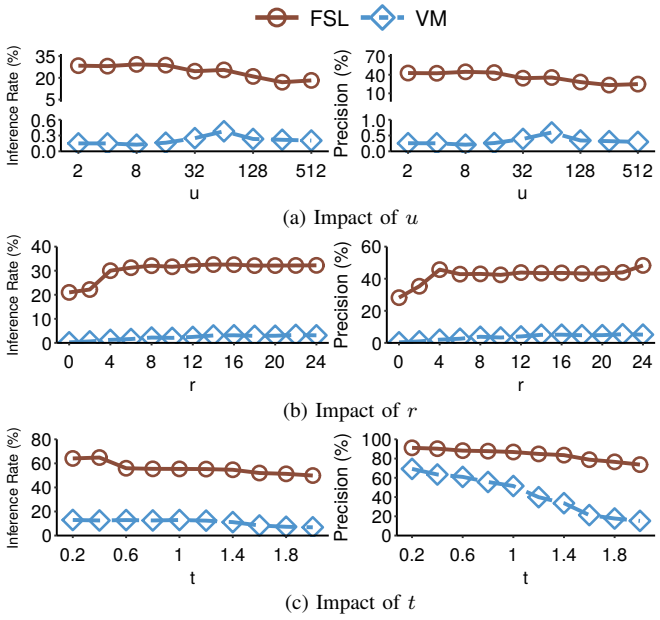


(a) FSL dataset: $g = 1$ (b) VM dataset: $g = 1$

(a) FSL dataset: $g = 2$ (b) VM dataset: $g = 2$

(a) FSL dataset: $g = 4$ (b) VM dataset: $g = 4$

Fig. 7. Attack using a sliding window (Exp#3).



(a) Impact of u

(b) Impact of r

(c) Impact of t

Fig. 8. Impact of u, r and t (Exp#4).

- It preserves high severity (e.g., the inference rate is above 50% with the precision above 85%), even when the auxiliary backup and the target backup are loosely correlated.

C. Impact of Parameters

Since the distribution-based attack is parameterized, we now study how the parameters affect the attack severity. For both FSL and VM, we fix the auxiliary information as the first backup, and aim to infer the plaintexts in the fifth backup.

Exp#4: Impact of u, r and t (Figure 8). 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 [22]). Figure 8(a) shows the impact of u, varied from 2 to 512. We have the same observation as in the locality-based attack [21] that the inference rate generally decreases, since more false positives are likely to be included in the inference results when u is large. Note that the prior work [21] does not report the inference precision, while we observe that the inference precision is fairly low (e.g., less than 45% for FSL and 0.6% for VM) and generally decreases with u.

Then, we fix $u = 128$ and $t \to \infty$, and evaluate the impact of r. Figure 8(b) shows the results when we vary r from 0 to 24. We observe that the inference rate generally increases with r from 21.0% to 32.2% for FSL and from 0.2% to 3.1% for VM, since the distribution-based attack now addresses disturbances to frequency ranking and infers more correct ciphertext-plaintext pairs. On the other hand, increasing r sometimes affects inference precision, since a large range of plaintexts are examined, thereby raising the possibility of introducing false positives. For example, when r is from 4 to 10 for FSL, the inference precision slightly drops from 45.7% to 42.4%.

Furthermore, we fix $u = 128$ and $r = 16$, and evaluate the impact of t. Figure 8(c) shows the results. When t is small, the attack misjudges and filters a number of inferred ciphertext-plaintext pairs, even they are correct. This introduces *false negatives*, thereby reducing the inference rate. For example, when t first increases (e.g., from 0.2 to 0.4 for FSL and to 1 for VM), the inference rate slightly increases (e.g., from 64.1% to 64.9% in FSL and from 12.5% to 12.9% in VM) for reducing false negatives. When t further increases to 2, the inference rate drops to 49.9% for FSL and 6.9% for VM, since a larger t cannot filter false positives effectively. For the same

TABLE II
SECURITY DAMAGE ON FILES (EXP#5): INFERENCE RATE AND PRECISION.

Ext. Name	Size	Common Rate	Unique Rate	Infer. Rate	Precision
.c	17.4KB	97.9%	23.9%	73.6%	85.0%
.h	6.0KB	94.9%	25.8%	73.5%	84.5%
.csv	13.6KB	95.5%	64.3%	83.3%	94.2%
.py	8.8KB	39.8%	53.6%	15.8%	64.8%
.cpp	6.7KB	98.3%	79.4%	76.7%	85.3%
.pdf	436.2KB	94.2%	46.9%	70.3%	89.1%
.jpg	450.5KB	97.6%	84.0%	2.8%	78.2%
.log	644.1KB	44.3%	39.6%	9.7%	83.7%
.vmdk	269.9MB	80.6%	42.4%	67.3%	90.9%
.so	878.7KB	47.6%	54.2%	40.5%	97.7%
.gz	54.4MB	99.7%	95.4%	6.5%	99.6%
.doc	76.1KB	99.2%	52.4%	58.6%	85.4%
.ppt	734.2KB	99.9%	52.2%	78.9%	88.7%
.pptx	1.27MB	91.5%	42.7%	77.2%	89.6%
.docx	73.9KB	99.1%	26.8%	77.1%	82.4%
.tgz	24.1MB	100.0%	98.9%	6.2%	99.8%

reason, the inference precision always decreases with t. Even so, filtering false positives ensures that both the inference rate (55.9% for FSL on average) and inference precision (84.2% for FSL on average) are significantly high.

Summary. We summarize our observations about the parameters' impact on the severity of the distribution-based attack. They can be used to guide the configuration of the distribution-based attack in practice.

- A relatively larger u helps increase the coverage of inferred ciphertext-plaintext pairs, yet decreasing both inference rate and precision.
- A relatively larger r provides more opportunities of identifying correct ciphertext-plaintext pairs, yet increasing the probability of having false positives.
- A smaller t filters a large fraction of false positives, yet introducing more false negatives.

D. File-level Damage

While previous experiments focus on inferring chunk-level plaintexts, we now study how the inferred chunks damage the security of files.

Exp#5: Security damage on files (Table II and Figure 9). We use the first FSL backup to infer the plaintexts in the fifth FSL backup, and evaluate the inference rate and precision *against different files*. Specifically, we now define the inference rate for a file type (i.e., the files that have the same extension name) as the number of correctly inferred plaintexts belonging to corresponding files divided by that of total unique plaintexts for these files. Similarly, the inference precision is the ratio of the number of correctly inferred plaintexts corresponding to a file type by that of all inferred plaintexts for the same type. Note that we focus on FSL, since only the FSL dataset includes the metadata (e.g., the extension name of each file).

Table II shows the attack results for some of the popular files [34] in FSL snapshots, and we also include the factors that potentially impact the attack severity. Specifically, for each extension name, we consider three factors: (i) *size*, which is the average size of corresponding files; (ii) *common rate*, which is the fraction of the common (unique) chunks of these files in the auxiliary and target backups; (iii) *unique rate*, which is the fraction of the unique chunks for corresponding files in the target backup. If an extension name has a high common rate, the corresponding files in the target backup incur only a few updates from the auxiliary backup. Also, if an extension name has a high unique rate, the chunks in the corresponding files in the target backup are more likely to be unique. Clearly, a high common rate and/or a low unique rate imply a more severe attack on the files. For example, about 78.9% of unique chunks in .ppt can be successfully inferred, while the compressed file format .tgz, whose unique rate is 98.9% only achieves the inference rate of 6.2%. Even so, we observe that the attack among all file types achieves high precision (e.g., 64.8-99.8%).

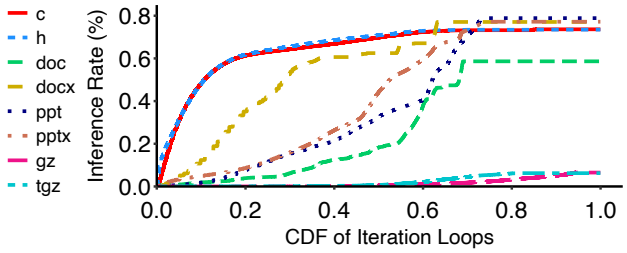


Fig. 9. Security damage on files (Exp#5): How different files are inferred in iterative inferences.

Figure 9 further shows the inference rates of some selected file types after each iteration of the distribution-based attack. The curves for .c and .h form similar shapes with the increase of the iteration loops, since corresponding files are likely to locate in identical programming directories and are inferred together. Also, the curves for .docx and doc, as well as for .pptx and .ppt increase in similar tendencies with the iteration loops. The possible reason is that the physical chunks of these files are organized in similar ways.

Summary. We present our observations about the file-level attack severity of the distribution-based attack.

- In addition to the correlation of the auxiliary files, the attack severity depends on how many unique chunks in the target files. The files with a low fraction of unique chunks are more vulnerable to be attacked.
- The files with similar file structures are likely to have similar inference patterns.

V. LIMITATIONS AND MITIGATIONS

Like the previous work [21], our distribution-based attack depends on the *locality* of chunks, and its severity should be degraded if the locality property is not preserved. However, our experiments show that the locality is generally prevalent in real-world backup workloads, while proactively breaking locality introduces additional metadata access overhead [21] and affects restore performance [37]. Another limitation is that any inference attack (e.g., [11], [17], [29]), including ours, depends on *the quality of the auxiliary information*. In Section IV-B, we show that the distribution-based attack preserves high severity even when the auxiliary backup is loosely correlated with the target backup (Exp#3).

In addition to breaking locality (that adds additional overhead [21], [37]), a natural way for defense is to *protect the frequency distribution of plaintexts*. Following this direction, MinHash encryption [21] and TED [23] break the one-to-one mapping of MLE and encrypt identical plaintexts into multiple distinct ciphertexts, in order to prevent an adversary from inferring the original frequencies of plaintexts. The cost of both approaches is *the degradation of storage efficiency*, since the storage system cannot remove all duplicate chunks by deduplication. Another paradigm for defense [2], [6] is to perform probabilistic encryption on each plaintext, and detect if the corresponding underlying plaintexts are identical via powerful cryptographic primitives (e.g., non-interactive zero knowledge [2] or fully homomorphic encryption [6]), which are not readily implemented in practice.

VI. Related Work

MLE instantiations. Recall from Section II that MLE [8] formalizes the cryptographic foundation of encrypted deduplication. The first published MLE instantiation is *convergent encryption (CE)* [12], which uses the cryptographic hash of a plaintext and its corresponding ciphertext as the MLE key and the tag, respectively. Other CE variants include: (i) *hash convergent encryption (HCE)* [8], which derives the tag from the plaintext while still using the hash of the plaintext as the MLE key; (ii) *random convergent encryption (RCE)* [8], 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 check; and (iii) *convergent dispersal (CD)* [24], 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 [7] 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).

To protect against the offline brute-force attack, DupLESS [7] implements server-aided MLE by managing MLE keys in a standalone key server, which ensures that each MLE key cannot be derived from a plaintext offline. Other studies extend server-aided MLE to address various aspects, such as reliable key management [13], transparent pricing [4], peer-to-peer key management [26], rekeying [30] and performance improvement [32]. However, server-aided MLE still builds on deterministic encryption and is vulnerable to the frequency analysis attack studied in this paper. Recently, S2Dedup [28] improves the security of MLE by computing the tags of chunks via trusted hardware. However, S2Dedup needs to detect duplicates by querying the index (based on the tags) maintained in unprotected memory, and still incurs the leakage channels exploited (see Section II-B) by our attack.

Attacks on encrypted deduplication and defenses. In addition to the offline brute-force attack, previous studies consider various attacks against deduplicated storage, and such attacks generally apply to encrypted deduplication as well. For example, the side-channel attack [15], [16] 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. The duplicate-faking attack [8] compromises message integrity via inconsistent tags. Ritzdorf *et al.* [33] exploit the leakage of chunk size to infer the existence of files. The locality-based attack [22] exploits frequency analysis to infer ciphertext-plaintext pairs. Our work follows the line of work on inference attacks [22], [33], yet provides a more in-depth study of inference attacks against encrypted deduplication with high inference precision.

Section V discusses the possible countermeasures against frequency analysis, yet they either degrade the storage saving of deduplication [21], [23] or incur high computational overhead [2], [6]. As mentioned before, server-aided MLE [7] defends against the offline brute-force attack. Server-side deduplication [16], [24] and proof-of-ownership [15] defend against the side-channel attack. Guarded decryption [8] defends against the duplicate-faking attack.

Other inference attacks. Several inference attacks have been proposed against encrypted databases (e.g., [29]) and keyword search (e.g., [11], [17]). Previous attacks focus on the plaintexts in just a small space that only includes hundreds or thousands of unique plaintexts, while we target the large-size space (e.g., on million level) in encrypted deduplication storage, and improve the inference precision.

VII. Conclusion

Encrypted deduplication applies deterministic encryption, and leaks the frequencies of plaintexts. This paper revisits the security vulnerability, and demonstrates that encrypted deduplication is even more vulnerable towards frequency analysis. We propose the distribution-based attack that simultaneously achieves high inference rate and inference precision. We empirically evaluate the attack with two real-world datasets, and present a variety of new observations about its natures. The source code of our attack implementations is now available at **http://adslab.cse.cuhk.edu.hk/software/freqanalysis**.

Acknowledgements

This work was supported in part by grants by National Natural Science Foundation of China (61972073), Key Research Funds of Sichuan Province (2021YFG0167), Sichuan Science and Technology Program (2020JDTD0007), Fundamental Research Funds for Chinese Central Universities (ZYGX2020ZB027, ZYGX2021J018), and CUHK Direct Grant 2020/21 (4055148).

References

[1] "Aol: Proudly releases massive amounts of private data," https://techcrunch.com/2006/08/06/aol-proudly-releases-massive-amounts-of-user-search-data/.

[2] M. Abadi, D. Boneh, I. Mironov, A. Raghunathan, and G. Segev, "Message-locked encryption for lock-dependent messages," in *Proc. of CRYPTO*, 2013.

[3] I. A. Al-Kadit, "Origins of Cryptology: The Arab Contributions," *Cryptologia*, vol. 16, no. 2, pp. 97–126, 1992.

[4] F. Armknecht, J.-M. Bohli, G. O. Karame, and F. Youssef, "Transparent data deduplication in the cloud," in *Proc. of ACM CCS*, 2015.

[5] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proc. of ACM CCS*, 2007.

[6] M. Bellare and S. Keelveedhi, "Interactive message-locked encryption and secure deduplication," in *Proc. of PKC*, 2015.

[7] M. Bellare, S. Keelveedhi, and T. Ristenpart, "DupLESS: Server-aided encryption for deduplicated storage," in *Proc. of USENIX Security*, 2013.

[8] ——, "Message-locked encryption and secure deduplication," in *Proc. of EUROCRYPT*, 2013.

[9] V. Bindschaedler, P. Grubbs, D. Cash, T. Ristenpart, and V. Shmatikov, "The tao of inference in privacy-protected databases," in *Proc. of VLDB*, 2017.

[10] J. Black, "Compare-by-hash: A reasoned analysis," in *Proc. of USENIX ATC*, 2006.

[11] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," in *Proc. of ACM CCS*, 2015.

[12] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system," in *Proc. of IEEE ICDCS*, 2002.

[13] Y. Duan, "Distributed key generation for encrypted deduplication: Achieving the strongest privacy," in *Proc. of ACM CCSW*, 2014.

[14] P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, and T. Ristenpart, "Leakage-abuse attacks against order-revealing encryption," in *Proc. of IEEE S&P*, 2017.

[15] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Proofs of ownership in remote storage systems," in *Proc. of ACM CCS*, 2011.

[16] D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Side channels in cloud services: Deduplication in cloud storage," *IEEE Security & Privacy*, vol. 8, no. 6, pp. 40–47, 2010.

[17] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation," in *Proc. of NDSS*, 2012.

[18] K. Jin and E. L. Miller, "The effectiveness of deduplication on virtual machine disk images," in *Proc. of ACM SYSTOR*, 2009.

[19] A. Juels and B. S. Kaliski, Jr., "Pors: Proofs of retrievability for large files," in *Proc. of ACM CCS*, 2007.

[20] M.-S. Lacharité and K. G. Paterson, "Frequency-smoothing encryption: Preventing snapshot attacks on deterministically encrypted data," *IACR Transactions on Symmetric Cryptology*, vol. 2018, no. 1, pp. 277–313, 2018.

[21] J. Li, P. P. C. Lee, C. Tan, C. Qin, and X. Zhang, "Information leakage in encrypted deduplication via frequency analysis: Attacks and defenses," *ACM Transactions on Storage*, vol. 16, no. 1, pp. 1–30, 2020.

[22] J. Li, C. Qin, P. P. C. Lee, and X. Zhang, "Information leakage in encrypted deduplication via frequency analysis," in *Proc. of IEEE/IFIP DSN*, 2017.

[23] J. Li, Z. Yang, Y. Ren, P. P. C. Lee, and X. Zhang, "Balancing storage efficiency and data confidentialitywith tunable encrypted deduplication," in *Proc. of ACM Eurosys*, 2020.

[24] M. Li, C. Qin, and P. P. C. Lee, "CDStore: Toward reliable, secure, and cost-efficient cloud storage via convergent dispersal," in *Proc. of USENIX ATC*, 2015.

[25] M. Lillibridge, K. Eshghi, D. Bhagwat, V. Deolalikar, G. Trezise, and P. Camble, "Sparse indexing: Large scale, inline deduplication using sampling and locality," in *Proc. of USENIX FAST*, 2009.

[26] J. Liu, N. Asokan, and B. Pinkas, "Secure deduplication of encrypted data without additional independent servers," in *Proc. of ACM CCS*, 2015.

[27] D. T. Meyer and W. J. Bolosky, "A study of practical deduplication," in *Proc. of USENIX FAST*, 2011.

[28] M. Miranda, T. Esteves, B. Portela, and J. Paulo, "S2dedup: SGX-enabled secure deduplication," in *Proc. of ACM SYSTOR*, 2021.

[29] M. Naveed, S. Kamara, and C. V. Wright, "Inference attacks on property-preserving encrypted databases," in *Proc. of ACM CCS*, 2015.

[30] C. Qin, J. Li, and P. P. C. Lee, "The design and implementation of a rekeying-aware encrypted deduplication storage system," *ACM Transactions on Storage*, vol. 13, no. 1, pp. 9:1–9:30, 2017.

[31] M. O. Rabin, "Fingerprinting by random polynomials," Center for Research in Computing Technology, Harvard University. Tech. Report TR-CSE-03-01, 1981.

[32] Y. Ren, J. Li, Z. Yang, P. P. C. Lee, and X. Zhang, "Accelerating encrypted deduplication via SGX," in *Proc. of USENIX ATC*, 2021.

[33] H. Ritzdorf, G. O. Karame, C. Soriente, and S. apkun, "On information leakage in deduplicated storage systems," in *Proc. of ACM CCSW*, 2016.

[34] Z. Sun, G. Kuenning, S. Mandal, P. Shilane, V. Tarasov, N. Xiao, and E. Zadok, "A long-term user-centric analysis of deduplication patterns," in *Proc. of IEEE MSST*, 2016.

[35] G. Wallace, F. Douglis, H. Qian, P. Shilane, S. Smaldone, M. Chamness, and W. Hsu, "Characteristics of backup workloads in production systems," in *Proc. of USENIX FAST*, 2012.

[36] B. Zhu, K. Li, and R. H. Patterson, "Avoiding the disk bottleneck in the data domain deduplication file system," in *Proc. of USENIX FAST*, 2008.

[37] X. Zou, J. Yuan, P. Shilane, W. Xia, H. Zhang, and X. Wang, "The dilemma between deduplication and locality: Can both be achieved?" in *Proc. of USENIX FAST*, 2021.