Revisiting Frequency Analysis against Encrypted Deduplication by Extending Locality

Jingwei Li, Jiacheng Liang, and Xiaosong Zhang

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 itself, 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 attacks that exploit the underlying storage workload characteristics. We propose the *distribution-based attack*, which extends locality to model the relative frequency distribution of ciphertexts and improves the inference precision of the previous attack. We evaluate the new attack against a real-world storage trace and provide insights into its actual damages.

Index Terms—Frequency analysis, chunk locality, similarity, encrypted deduplication, cloud storage

1 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 all already stored data copies. Field studies show that deduplication reduces substantial fractions of storage space under real-world storage workloads, such as backups [27], file system snapshots [20], and virtual disk images [14]. Cloud storage services (e.g., Dropbox and Google Drive) also use deduplication to save storage costs [12]. To ensure data confidentiality, clients should encrypt their own data before writing the data to a storage system. 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)* [6], whose instantiations have been extensively studied and realized in the literature (Section 2). MLE builds on *deterministic encryption*, which deterministically maps a plaintext to the same ciphertext to preserve the identical content pattern after encryption.

However, the deterministic nature of encrypted deduplication causes information leakage of the ciphertexts, such that the adversaries can analyze the ciphertexts and infer their original plaintexts. Specifically, an adversary can perform *frequency analysis* [2] on both the ciphertexts that are observed and the plaintexts that are known a 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 [21] and searchable keywords [8], [13] that are protected by deterministic encryption. Recently, frequency analysis is also shown effective against encrypted deduplication [16].

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, called the *distribution-based attack*, against encrypted deduplication. It target *backup workloads* [27] and exploits the *locality* [18], [29], which states that the ordering of data is likely preserved across various backup files. It aims to infer the ciphertext-plaintext pairs that are in similar positions of backup files, by examining the relative frequency distribution based on the co-occurrence frequencies of adjacent data to filter any falsely inferred ciphertext-plaintext pairs.

We conduct extensive trace-driven evaluation on a real-world backup storage workload. ...

2 Background

We focus on *chunk-based deduplication*, which operates at the granularity of small-size data units called *chunks*. Figure 1 summarizes the deduplication workflow. A storage system first partitions a file (e.g., backup) of a client into either

• *J. Li, J. Liang, and X. Zhang are with the University of Electronic Science and Technology of China, China (Emails: {jwli, johnsonzxs}@uestc.edu.cn, xx@gmail.com).*

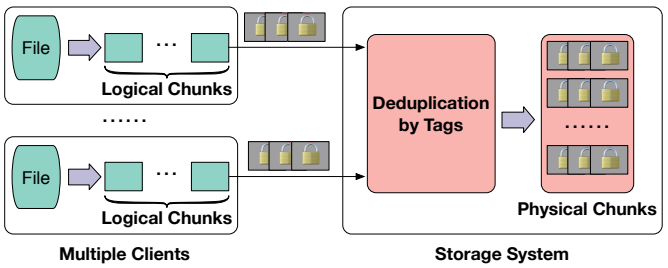


Fig. 1. Overview of deduplication workflow.

fixed-size or variable-size *logical chunks*; note that variable-size chunking identifies chunk boundaries in a content-dependent manner (e.g., via Rabin fingerprinting [23]) and is robust against content shifts [29]. 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 [7]. The storage system stores only a copy of identical logical chunks as a *physical chunk*, and each identical logical chunk refers to the same physical chunk via small-size references.

Encrypted deduplication addresses chunk confidentiality in an outsourcing environment (e.g., cloud storage), while preserving deduplication effectiveness. Traditional symmetric encryption requires that multiple clients encrypt their file data by their (distinct) secret keys, thereby converting identical chunks into distinct encrypted chunks that can no longer be deduplicated. *Message-locked encryption (MLE)* [6] formalizes encrypted deduplication as a cryptographic primitive, such that a symmetric key (called the *MLE key*) is derived from the content of a *plaintext* (e.g., a logical chunk in chunk-based deduplication), and encrypts the plaintext using the MLE key to form a *ciphertext* (e.g., an encrypted logical chunk). Thus, MLE ensures that identical plaintexts are always encrypted to identical ciphertexts. Finally, the storage system derives the tag from each ciphertext and performs deduplication.

MLE has been widely analyzed and realized in different storage environments (Section 6). Existing MLE instantiations mainly build on the notion called *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.

3 Threat Model

We model an original (i.e., unencrypted) file as an ordered list of *logical plaintexts* (i.e., logical chunks before encryption and deduplication), denoted by $\mathbf{M} = \langle \hat{M}^{(1)}, \hat{M}^{(2)}, \ldots \rangle$. Each logical plaintext $\hat{M}^{(i)}$ ($i \geq 1$) is encrypted into the corresponding ciphertext $\hat{C}^{(i)}$ via MLE, and all encryption results of $\mathbf{M}$ form a stream of *logical ciphertexts* $\mathbf{C} = \langle \hat{C}^{(1)}, \hat{C}^{(2)}, \ldots \rangle$. Note that the logical ciphertexts in $\mathbf{C}$ are *ordered* to reflect the deduplication processing sequence of an encrypted deduplication storage system. In addition, identical 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 into the corresponding unique ciphertext C via MLE. Each M (resp. C) corresponds to one or multiple identical copies of logical plaintexts $\hat{M}^{(i)}$ (resp. logical ciphertexts $\hat{C}^{(i)}$).

3.1 Adversarial Assumptions

We study the security of encrypted deduplication when an adversary passively monitors the stream of logical ciphertexts $\mathbf{C}$ being written to the storage system. For example, the adversary compromises the deduplication process (see Figure 1), identifies the characteristics of ciphertexts, and learns any leaked information about the corresponding plaintexts. We focus on two leakage channels in this paper.

- **Frequency:** Due to the deterministic nature of 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:** Some storage systems (e.g., [29]) apply deduplication to the chunks in the same order as they appear in the original file. Thus, the order of the ciphertexts in $\mathbf{C}$ reflects the order of the plaintexts in $\mathbf{M}$.

We assume that the adversary has access to some *auxiliary information* that presents the ground truth about the characteristics correlated with $\mathbf{M}$. Note that the availability of auxiliary information is necessary for any inference attack [21]. In this paper, we consider the auxiliary information as an ordered list of previously known plaintexts (e.g., old user backups), denoted by $\mathbf{A}$. Clearly, the attack severity 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 in encrypted deduplication.

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, which can be independently prevented by existing defense mechanisms. For example, proof-of-ownership [11] or enforcing server-side deduplication [12], [17] can prevent a malicious client from claiming the ownership of unauthorized files in client-side deduplication [12]; remote integrity checking [4], [15] protects modification of data in cloud storage.

3.2 Adversarial Goals

The adversary aims to infer the ciphertext-plaintext pairs, denoted by $\{(C, M)\}$, given the leakage of ciphertexts in $\mathbf{C}$ and the available auxiliary information in $\mathbf{A}$, with two key 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 the inferred ciphertext-plaintext pairs (i.e., high precision or low false positive rates in statistical terms).

We focus on frequency analysis as the main attack methodology, which infers ciphertext-plaintext pairs against deterministic encryption. Classical frequency analysis [2] sorts the unique ciphertexts in $\mathbf{C}$ and the unique plaintexts in $\mathbf{A}$ by frequency. It then relates each unique ciphertext in $\mathbf{C}$ with the unique plaintext in $\mathbf{A}$ that has the same frequency rank. In this paper, we exploit the locality property to design more severe frequency analysis against encrypted deduplication, with a high inference rate and a high inference precision.

4 Distribution-based Attack

We present the distribution-based attack, which leverages the order leakage of $\mathbf{C}$ and $\mathbf{A}$ to launch frequency analysis. The distribution-based attack builds on the locality property [18], [29] of practical backup workloads. It extends the previously proposed locality-based attack [16] by modeling the relative frequency distributions of $\mathbf{C}$ and $\mathbf{A}$, so as to filter false positives and achieve a high inference precision.

4.1 Background: Locality-based Attack

We first provide the background of the locality-based attack [16], and then highlight its major weakness.

The locality-based attack targets periodic backups that are created as the complete copies of primary data (e.g., application states, file system snapshots, and virtual disk images) on a daily or weekly basis. It assumes that the auxiliary information $\mathbf{A}$ is derived from some older backups, and its goal is to infer the ciphertext-plaintext pairs from the latest backup (i.e., $\mathbf{M}$).

The locality-based attack leverages *locality*, which is commonly found in practical backup workloads [18], [29]. 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 of each unique ciphertext C and its *co-occurrence frequency* with any of its left and right neighbor (e.g., $\hat{C}^{(i-1)}$ and $\hat{C}^{(i+1)}$). 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 (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.

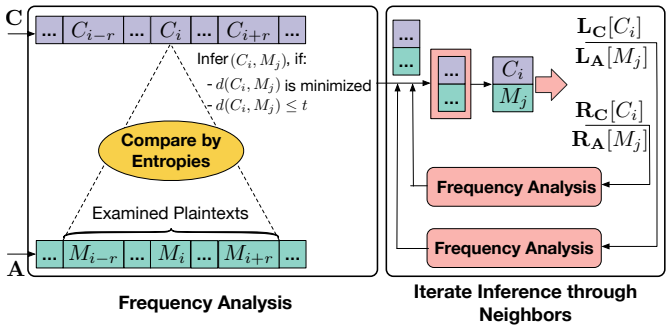


Fig. 2. Workflow of distribution-based attack.

However, the locality-based attack has a major weakness that it can introduce a high number of false positives (i.e., incorrect 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 incorrect ciphertext-plaintext pairs and in turn compromise the inference of their neighbors. Although the locality-based attack is shown to effectively infer a significant fraction of correct ciphertext-plaintext pairs (i.e., high inference rate) [16], 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).

4.2 Description of Distribution-based Attack

The distribution-based attack extends the locality-based attack to remove false positives. It also leverages locality as in the locality-based attack. In addition, for each unique ciphertext C in $\mathbf{C}$, the distribution-based attack measures its *relative frequency distribution* based on its co-occurrence frequencies with its neighbors. Similarly, it also measures the relative frequency distribution of each unique plaintext M in $\mathbf{A}$. Our observation is that for a correct ciphertext-plaintext pair (C, M), both C and M should have similar relative frequency distributions; that is, their co-occurrence frequencies with their respective neighbors are similar. We treat this observation as a generalized notion of locality and use it as a condition to filter possibly incorrect ciphertext-plaintext pairs.

Figure 2 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 locality-based attack, it configures the parameter u, so the underlying frequency analysis returns at most u ciphertext-plaintext pairs. For each unique ciphertext C_i of rank i ($1 \leq i \leq u$), 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., 10 by default) that is used to address the range of rank disturbance. 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 4.1).

For each C_i ($1 \leq i \leq u$) and the corresponding M_j ($i - r \leq j \leq i + r$), the distribution-based attack compares their relative frequency distributions by *entropy*, which measures the amount of information produced by a random source. Here, we use the entropy to *quantify the randomness of probability distribution* [28]. Specifically, the attack defines two random variables, 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, while the event "$Y = C$" denotes the case that C is the right neighbor of C_i. The attack computes 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 neighbor C' and its right neighbor C'', respectively. Both X and Y follow the relative frequency distributions of C_i, and their randomness can be characterized by entropies, denoted by $e(\mathbf{L_C}[C_i])$ and $e(\mathbf{R_C}[C_i])$, respectively, as follows:

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

The attack also computes 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. It compares the relative frequency distributions of C_i and M_j by the *Euclidean distance*, denoted by $d(C_i, M_j)$:

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

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 $i - r \leq j \leq i + r$.
- **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) such that 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. 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.

The distribution-based attack follows the iteration paradigm in the locality-based attack (Section 4.1) 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 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 range of rank disturbance to be addressed, and (iii) t, which specifies the Euclidean distance threshold to filter possibly incorrect inference results. The prior locality-based attack 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).

5 Attack Evaluation

We evaluate the distribution-based attack (Section 4) based on the FSL dataset. Specifically, the FSL dataset is collected by the File systems and Storage Lab at Stony Brook University [26]. We focus on the *fslhomes*, which contains the snapshots of students' home directories from a shared network file system. Each snapshot is represented as an ordered list of file metadata (e.g., full pathname), as well as the corresponding 48-bit chunk fingerprints obtained from variable-size chunking. We consider the snapshots, whose chunks have an average chunk size of 8 KB. 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.

Note that our dataset does not contain actual content, so we mimic the adversarial knowledge based on chunk hashes. Specifically, we select 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. We evaluate the attack effectiveness by the inference rate and the inference precision (defined in Section 3.2).

5.1 Attack Effectiveness

Exp#1 (Overall attack effectiveness). We evaluate the overall attack effectiveness of the distribution-based attack, in

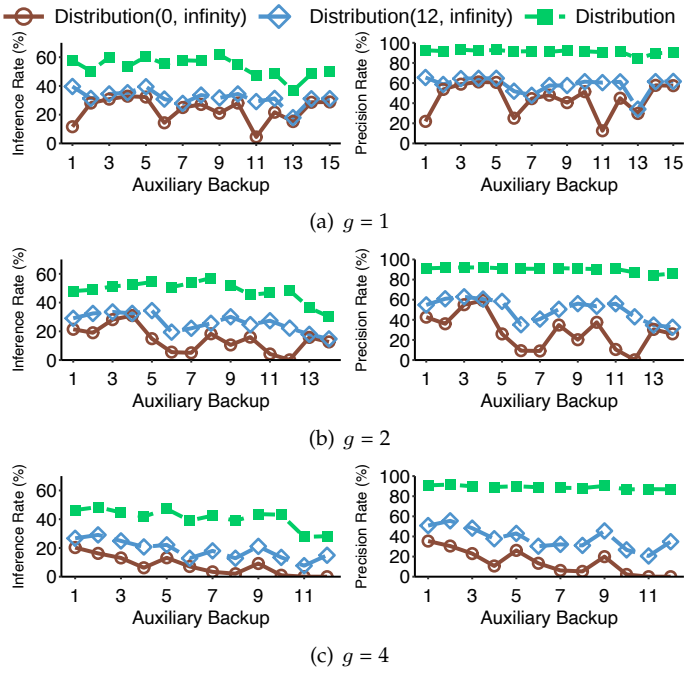
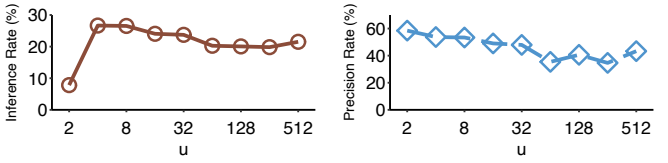


Fig. 3. Exp#1 (overall attack effectiveness).

order to justify the necessity of r (that is used to address disturbance to frequency ranking) and t (that is used to filter out unreasonable inference results). Specifically, we consider three distribution-based attack instances: (i) $\texttt{distribution}(0, \infty)$, which does not address any disturbance to the frequency ranking; (ii) $\texttt{distribution}(12, \infty)$, which addresses the rank disturbance within a range of 12 but does not filter out any inference results; and (iii) `distribution`, which applies our default configurations. Note that the first instance $\texttt{distribution}(0, \infty)$ is equivalent to the previous attack [16]. We launch the considered attacks based on a sliding window approach. Specifically, we choose the i-th backup as the auxiliary information, and infer the original plaintext chunks in the $(i + g)$-th backup, while we vary i and g in our evaluation (the smaller g is, the more correlated are the auxiliary and the target backups are).

Figure 3 shows the results for both inference rate and precision. We observe that `distribution` achieves more severe than its two special instances. For example, when we configure g = 1, 2 and 4, it achieves the average inference rates of 53.3%, 48.3% and 41.0%, respectively, significantly higher than those of $\texttt{distribution}(0, \infty)$ (23.3%, 14.5% and 7.6%) and $\texttt{distribution}(12, \infty)$ (31.9%, 26.1% and 18.7%). The reason is that `distribution` can find the likely correct inferred pairs in a ranking range, as well as filter out some unreasonable inferred pairs. This also implies a high inference precision for `distribution`. For example, the average inference precisions of `distribution` are 91.2%, 90.0% and 89.0% for g = 1, 2 and 4, respectively, while those of $\texttt{distribution}(0, \infty)$ (44.5%, 28.4% and 14.4%) and $\texttt{distribution}(12, \infty)$ (58.2%, 50.0% and 38.0%) are below 60%.

Observation (1) – *By mitigating ranking disturbance and filtering unreasonable results, we can improve the inference rate and precision of the previous attack [16] by up to 5.4× and 6.2×, respectively.*



(a) Inference rate (b) Inference precision

Fig. 4. Exp#2 (Impact of u).

In addition, with a smaller g, all attacks achieve relatively more severe, since the auxiliary backup is correlated with the target backup. However, when we increase g, the attack effectiveness drops, sharply for $\texttt{distribution}(0, \infty)$. For example, when we increase g from 1 to 4, its inference rate and precision drop by 67.2% and 209.7%, respectively. The reason is that $\texttt{distribution}(0, \infty)$ is sensitive to the correlation between the auxiliary and the target backups. On the other hand, the corresponding degradation ratios of `distribution` are only 23.1% and 2.4%, respectively.

Observation (2) – *Even when the auxiliary backup is loosely correlated with the target backup, mitigating ranking disturbance helps preserve high inference rate (e.g., reducing the corresponding degradation ratio of the previous attack by 2.9×) and filtering unreasonable inferred results helps preserve high inference precision (e.g.,reducing the corresponding degradation ratio of the previous attack by 87.6×).*

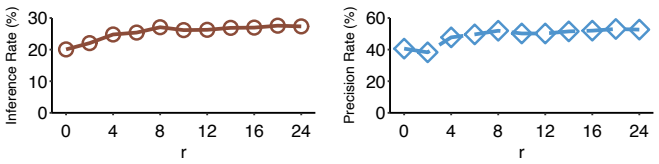
We find a phenomenon for $\texttt{distribution}(0, \infty)$, in which a more correlated auxiliary backup sometimes leads to lower attack effectiveness. For example, when we use the first backup as the auxiliary backup, the inference rates of $\texttt{distribution}(0, \infty)$ are 11.6%, 21.4% and 20.3% for g = 1, 2 and 4, respectively. The possible reason is that the attack iterates on some incorrect inferred results initially, and hence leads to the compromises of the following inference results (e.g., the inference precision of g = 1 is as low as 21.9%). This also happens on $\texttt{distribution}(12, \infty)$ when it uses the 2nd backup as the auxliary information, but we find that `distribution` is robust against the compromised inference.

5.2 Impact of Parameters

Next, we study how the parameters affect the attack effectiveness of the distribution-based attack. Here, we fix the auxiliary information as the 1st backup, and aim to infer the original chunks in the 5-th backup.

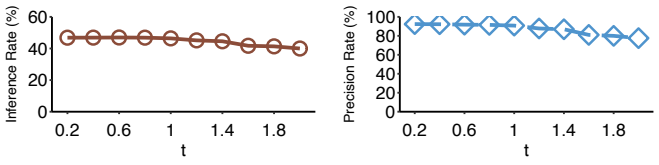
Exp#2 (Impact of u). We configure $t \to \infty$ and $r = 0$ to evaluate the impact of u (in this case, the distribution-based attack reduces to the locality-based attack [16]).

Figure 4(a) shows the impact of u, varied from 2 to 512. We have the same observation as in the locality-based attack [16] that the inference rate first increases and then decreases with u. Note that the prior work [16] does not report the inference precision, while we observe that the inference precision is at fairly low (e.g., less than 60%) and generally decreases with u. The reason is that the false positives are more likely to be included (in the inference results) when u is large.



(a) Inference rate (b) Inference precision

Fig. 5. Exp#3 (Impact of r).



(a) Inference rate (b) Inference precision

Fig. 6. Exp#4 (Impact of t).

Observation (3) – *A relatively larger u increases the inference rate, yet decreasing the inference precision.*

Exp#3 (Impact of r). Then, we fix u =64 and $t \to \infty$, and evaluate the impact of r. Figure 5 shows the results when we vary r from 0 to 24. We observe that the inference rate increases with r from 20.1% to 27.2%, since the distribution-based attack addresses disturbances to frequency ranking and infers more correct ciphertext-plaintext pairs. On the other hand, a too large r affects inference precision, since a large range of plaintexts are examined, increasing the possibility of introducing false positives. For example, when we increase r to 24, it drops from 53% to 52.6%. We also the inference precision when r is beyond 24, and the precision finally drops to 45.4% for r = 32.

Observation (4) – *A larger r provides more opportunities of identifying correct ciphertext-plaintext pairs, yet increasing the probability of having false positives.*

Exp#4 (Impact of t). Furthermore, we fix u = 64 and t = 12, and evaluate the impact of t. Figure 6 shows the results. When t is small (e.g., less than 0.6), the attack misjudges and filters a number of ciphertext-plaintext pairs, even they are correct. This introduces *false negatives*, thereby reducing the inference rate. For example, when t first increases from 0.2 to 0.6, the inference rate slightly increases from 46.8% to 47.0% for reducing false negatives. When t further increases to 2, the inference rate drops to 40.0%, since a larger t cannot filter false positives effectively. For the same reason, the inference precision always decreases with t. Even so, filtering false positives ensures that both the inference rate (above 40%) and inference precision (above 77%) are at significantly high.

Observation (5) – *A smaller t filters a large fraction of false positives, yet introducing more false negatives.*

5.3 Case Study (Need Additional Experiment)

We evaluate the security implication of the distribution-based attack via a case study. We launch the attack over files and examine its damage on different types of files.

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 **??**(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 **??**(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.*

6 Related Work

MLE instantiations: Recall from Section 2 that MLE [6] formalizes the cryptographic foundation of encrypted deduplication. The first published MLE instantiation is convergent encryption (CE) [9], 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) [6], which derives the tag from the plaintext while still using the hash of the plaintext as the MLE key; random convergent encryption (RCE) [6], 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) [17], 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 [5] 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 [5] 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 [10], transparent pricing

[3], peer-to-peer key management [19], and rekeying [22]. However, server-aided MLE still builds on deterministic encryption. To our knowledge, existing MLE instantiations (based on CE or server-aided MLE) are all vulnerable to the inference attacks studied in this paper.

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 [11], [12] 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 [6] compromises message integrity via inconsistent tags. Ritzdorf *et al.* [24] exploit the leakage of chunk size to infer the existence of files. The locality-based attack [16] exploits frequency analysis to infer ciphertext-plaintext pairs. Our work follows the line of work on inference attacks [16], [24], yet provides a more in-depth study of inference attacks against encrypted deduplication with high inference precision and/or low adversarial assumption.

Defense mechanisms: MinHash encryption [16] defeats frequency analysis by breaking the one-to-one mapping of MLE, yet it degrades the storage saving of deduplication. Other defense mechanisms (e.g., [25]) against frequency analysis are computationally expensive, and how they can be deployed in practice remains unexplored. As mentioned before, server-aided MLE [5] defends against the offline brute-force attack. Server-side deduplication [12], [17] and proof-of-ownership [11] defend against the side-channel attack. Server-side tag generation [5], [9] and guarded decryption [6] defend against the duplicate-checking attack.

Other inference attacks: Several inference attacks have been proposed against encrypted databases (e.g., [21]) and keyword search (e.g., [8], [13]). 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.

7 Conclusion

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: the distribution-based attack achieves high inference rate and precison, while the clustering-based attack relies on weak assumptions of adversarial knowledge. We empirically evaluate our attacks with three real-world datasets, and present a variety of new observations about their natures.

Our attacks build on the characteristics of practical deduplication workloads. While such characteristics has been exploited to improve deduplication performance and storage efficiency in practice, we show how to exploit it from an adversarial perspective. We hope that our findings help practitioners and researchers better understand the vulnerabilities in encrypted deduplication and motivate more research along this direction.

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

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

[4] 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.

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

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

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

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

[9] 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.

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

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

[12] 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.

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

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

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

[16] 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.

[17] 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.

[18] 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.

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

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

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

[22] 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.

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

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

[25] J. Stanek, A. Sorniotti, E. Androulaki, and L. Kencl, "A secure data deduplication scheme for cloud storage," in *Proc. of FC*, 2014.

[26] 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.

[27] 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.

[28] Q. A. Wang, "Probability distribution and entropy as a measure of uncertainty," *Journal of Physics A: Mathematical and Theoretical*, vol. 41, no. 6, 2008.

[29] 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.