# Side Channels in Deduplication: Trade-offs between Leakage and Efficiency

Frederik Armknecht
University of Mannheim
Mannheim, Germany
armknecht@uni-mannheim.de

Colin Boyd
Norwegian University of Science and Technology, NTNU
Trondheim, Norway
colin.boyd@item.ntnu.no

Gareth T. Davies
Norwegian University of Science and Technology, NTNU
Trondheim, Norway
gareth.davies@ntnu.no

Kristian Gjøsteen
Norwegian University of Science and Technology, NTNU
Trondheim, Norway
kristian.gjosteen@math.ntnu.no

Mohsen Toorani
University of Bergen
Bergen, Norway
mohsen.toorani@uib.no

## ABSTRACT

Deduplication removes redundant copies of files or data blocks stored on the cloud. Client-side deduplication, where the client only uploads the file upon the request of the server, provides major storage and bandwidth savings, but introduces a number of security concerns. Harnik et al. (2010) showed how cross-user client-side deduplication inherently gives the adversary access to a (noisy) side-channel that may divulge whether or not a particular file is stored on the server, leading to leakage of user information. We provide formal definitions for deduplication strategies and their security in terms of adversarial advantage. Using these definitions, we provide a criterion for designing good strategies and then prove a bound characterizing the necessary trade-off between security and efficiency.

## 1. INTRODUCTION

Deduplication is a process used by many cloud storage providers and services to remove redundant copies of data stored in the cloud. It has been shown [12, 14] to greatly reduce storage requirements in practice because users, both individuals and corporations, often store identical or similar content. Deduplication can take place either at the server-side or at the client-side. In *server-side deduplication*, the server checks whether a file uploaded by a client has already been stored. If so, the server does not store it again but instead records the ownership by the client and allows the client to access the shared file using a suitable index. Server-side deduplication achieves the aim of reducing storage but still requires the client to upload each file it wishes to store. In *client-side deduplication*, a user wishing to upload a file first checks whether the file is already stored in the cloud, for example by sending a hash of the file to the server which checks against its list of stored file hashes. If the file is already stored then the file is not sent by the client, but the server allows the client access to the shared file as before. Thus client-side deduplication greatly reduces the bandwidth requirements in cloud storage in addition to reducing storage requirements. Since communication costs can be high in comparison with storage costs, client-side deduplication is generally preferable to server-side deduplication on economic grounds. Deduplication can take place either with respect to files or with respect to blocks, but we will not be concerned with this difference since many of the attacks and countermeasures considered in this paper can be applied to either approach.



*ASIA CCS '17, April 02-06, 2017, Abu Dhabi, United Arab Emirates*



DOI: http://dx.doi.org/10.1145/3052973.3053019

### *Secure Deduplication.*

Despite the great saving in storage and bandwidth, deduplication causes at least two major security and privacy problems, and this has led to extensive recent work on *secure deduplication* [14]. The first problem is that deduplication cannot take place if semantically secure end-to-end encryption is deployed. Under ciphertext indistinguishability the cloud service provider (CSP), which does not possess the decryption key, would be unable to determine if two ciphertexts correspond to the same plaintext. Several alternative forms of encryption have been proposed in order to address this problem [1,3,5,8,9,18], deriving the encryption key from the file itself in various ways. These works often make strong assumptions regarding file unpredictability or key distribution. The second problem is that client-side deduplication can work as a side channel leaking information under different attacks [7]. This paper focuses on these side-channel attacks.

Harnik et al. [7] identified three attacks due to side-channels in client-side deduplication. The attacks apply to the cross-user scenario where different users who upload the same file will have their data deduplicated. The basic idea of all the attacks is that one user can obtain information about an-

other user's file by receiving a signal revealing whether or not the file was previously uploaded. In one example, sometimes called the *salary attack*, the adversary attempts to learn private data of Alice (her salary) in her employment contract which she has stored with the CSP. The adversary inserts guesses on a template file and uploads it to the CSP where the corresponding file of Alice resides. Occurrence of the deduplication signal will then allow the adversary to infer correctness of the guess.

Having identified these side-channel attacks, Harnik et al. [7] proposed a countermeasure in which the signal on whether a file is already uploaded is hidden by randomization. More specifically, for each file a threshold is chosen uniformly at random and the user is only informed not to upload the file if the number of previous uploads meets or exceeds the threshold. This will obviously increase the required bandwidth compared to basic client-side deduplication. The side-channel does not occur in server-side deduplication because then the client always uploads the file and allows the server to decide whether or not to deduplicate. The random threshold countermeasure can thus be seen as a compromise between the efficiency of client-side deduplicaton and the security of server-side deduplication.

### *Contributions.*

Although the mitigation idea of Harnik et al. [7] has been discussed and developed in the literature [10, 17], there has been no formal modeling and analysis of threshold-based solutions for defending side-channel attacks. This has prevented any opportunity to formally compare the effectiveness of different solutions. The purpose of this paper is to remedy this situation by:

- providing formal definitions for side-channel deduplication strategies, including a natural measure for effectiveness of countermeasures;
- identifying the conditions required for strategies to optimize bandwidth and security;
- characterizing the trade-off between security and efficiency necessary for all strategies;
- showing that the original proposal of Harnik et al. [7] provides an optimal defence within one natural security measure.

There are other scenarios in which similar kinds of side-channels are available to attackers. In independent and concurrent work, Ritzdorf et al. [16] consider the information leaked to a curious cloud provider in deduplicating storage systems, with particular focus on leakage caused by using content-defined chunking as the segmentation mechanism. They show empirically that under a number of strong assumptions on the target files, a cloud provider can infer the contents of low-entropy files with high probability even if the encryption key is unknown. This attack vector is tangential to the problem tackled in this paper, but it does emphasize the need for rigour in analyzing security of cloud storage in the presence of malicious clients and servers. Another closely related area is cache privacy attacks such as those considered by Ács et al. [2] in Named Data Networking; we believe that our model can also be applied to such scenarios.

The rest of this paper is organized as follows. Side-channel attacks on cloud storage and some existing countermeasures are reviewed in Section 2. Our security model and optimality of defences are discussed in Section 3. Section 4 proves our main theorems relating security and efficiency, characterizing both good dedpulication strategies and the essential trade-off between security and efficiency. In Section 5 we discuss how our work relates to other countermeasures and approaches.

## 2. SECURITY FOR DEDUPLICATION

This section reviews the current status of side-channel attacks on client-side deduplication and their countermeasures. For the rest of this paper, we will discuss client-side deduplication only, unless explicitly stated otherwise, and focus only on side-channel issues. We define *users* as the entities with distinct logins to a system, and *clients* as the devices that interact with the server on behalf of their owner, the user. Users and clients may be adversarially controlled: for the attacks we describe we consider an adversary that has access (i.e. login credentials) to the cloud storage service and attempts to glean information from its interactions with the server. The side-channel attacks we focus on are not the only type of attack in this scenario. If files can be retrieved using only a (deterministically-derived) index such as the hash of the file then this introduces the issue of users being able to share files with others, potentially creating copyright issues [13]. This issue can be solved by incorporating proofs of ownership (PoW) [6] into the deduplication process.

### 2.1 Existence-of-File Side-Channel Attack

Harnik et al. [7] identified the side channel inherent in client-side deduplication and discussed its implications in terms of three closely-related attacks performed by an adversary that follows the upload protocol correctly.

1. *Learning file contents.* An attacker can guess the contents of a file and infer its existence in the cloud.
2. *Identifying files.* The adversary can identify whether an incriminating file that should not be in the cloud, such as pirated media or a leaked document, is stored. If found, the owner could be later identified with the help of law enforcement access.
3. *Covert channels.* The existence of a unique file in the cloud can be used to signal a bit in a covert communication channel.

These are three outcomes of the same attack mechanism: an adversary wishes to learn whether or not a file has previously been uploaded to the storage of a CSP and then does something with the single bit of information it learns. We will therefore use the general term *existence-of-file attack* to incorporate any attack in which the adversary aims to learn whether or not a file has been previously uploaded. This term includes the notion of the aforementioned *salary attack* because of the following scenario.

- In order to implement client-side deduplication, the client first sends a short identifier to the CSP. The CSP instructs the client to upload the full file only if it is not already stored in the cloud.
- The adversary creates a template of an employment contract of Bob and attempts a number of uploads of files that only differ in a specific field (e.g. the salary).

- At some point, the upload will be halted by the CSP. The adversary will then learn that this file is already stored on the cloud and that her guess on Bob's salary is correct.

Examples of other sensitive information that an adversary may like to learn via this attack vector are clinical lab test results, figures in tax returns, pay stubs and contracts, and bank letters including a password or PIN. Note that these attacks are not just an issue if the files are unencrypted, they also apply if the files are encrypted using a method that allows the server to learn equality of underlying plaintexts, for example by using a key that is deterministically derived from the file [3, 5, 9, 11].

In Section 3, we will formalize deduplication and give our security definition for the existence-of-file attack. There are subtleties in the desired outcome of the attack: does the adversary want to know the answer to "Is Bob's salary X?" or "What is Bob's salary?" We address these issues and the challenges in formally modeling this scenario later on.

In the next section and in the rest of the paper, we will describe a countermeasure used to negate the effects of these side-channel attacks while still allowing client-side deduplication. We note that other approaches may also be used to counteract these attacks. For example, the server could ask clients to separate all files at the point of upload into *sensitive* or *non-sensitive*: files with the flag *sensitive* are encrypted using semantically-secure encryption before they are uploaded, and others are uploaded normally. However most users will simply bypass this step by marking all files with one of the flags: This either increases cost for the CSP by preventing deduplication or leaves the files vulnerable to attack. Any such countermeasure that requires the user to make decisions about their files is unrealistic in practice.

## 2.2 Randomized Solution of Harnik et al.

An approach to defending against the side-channel attacks is to require users to upload files even in the case that they have previously been already uploaded. For a given file, denote as $\mathsf{thr}$ the number of uploads before the server informs clients that it has enough copies. When a user chooses to store the same file, the server checks whether the $\mathsf{thr}$ is reached for that file and if not requires the file to be uploaded and increments the counter. Any strategy for which $\Pr[\mathsf{thr} = 1] \neq 1$ for all files will impose increased bandwidth until the threshold is reached. In addition, if the adversary knows the threshold $\mathsf{thr}$ for a given file then she can count the number of uploads allowed and still infer whether the file initially existed depending on whether she is required to upload $\mathsf{thr}$ times, or $\mathsf{thr} - 1$ times. If this is the case then the classic attack is just slowed down. Consequently we assume that adversary $\mathcal{A}$ does not know the value of $\mathsf{thr}$ (which would differ per file) but $\mathcal{A}$ may know how $\mathsf{thr}$ is selected.

Harnik et al. [7] proposed use of a randomized threshold for each file, and this approach has since been adopted by Liu et al. [11]. Their intuition was as follows: if $\mathsf{thr}$ is chosen uniformly from the range $\{1, \ldots, \mathsf{B}\}$ for some integer $\mathsf{B}$ then an adversary launching the existence-of-file attack will learn nothing if $\mathsf{thr} \in \{2, \ldots, \mathsf{B} - 1\}$. For the rest of the paper this value $\mathsf{B}$ is the upper bound for the threshold. Note that for this approach, the expected number of uploads of a file is $\frac{\mathsf{B}+1}{2}$.

Note that if the system does not attempt to defend against the side-channel attacks, the first uploader of a given file will be required to upload but all subsequent uploads will not be required. This corresponds to the case $\mathsf{B} = 1$ which is then basic client-side deduplication and is optimal in terms of bandwidth usage. In contrast, if the system wishes to leak no information then the server will always require upload of each file, which will of course incur a significant bandwidth cost. This corresponds to an infinite $\mathsf{B}$ which is equivalent to server-side deduplication. Thus from an efficiency point of view using a finite threshold is considerably better than negating the attacks using server-side deduplication.

The interesting cases are where $\mathsf{B}$ is finite and $\mathsf{B} > 1$. If $\mathsf{thr} = 1$ and on the first upload $\mathcal{A}$ is not asked to upload the file, then $\mathcal{A}$ will learn that the file was already stored. Likewise, if $\mathcal{A}$ is asked to perform $\mathsf{B}$ uploads then she will learn that the file was certainly not already stored. We can see a clear tradeoff between security and efficiency since $\mathsf{B}$ indicates the maximum number of times a file may need to be uploaded and is thus the worst-case overhead for bandwidth.

A uniformly random choice of $\mathsf{thr}$ is an intuitively reasonable option for defending against the side-channel attacks. However, it is not the only option. Even for a fixed upper bound $\mathsf{B}$, it is not immediate that a uniform probability distribution is best for security. When taking into account the trade-off between security and efficiency, the question becomes more complex. The threshold $\mathsf{thr}$ could be chosen according to some other probability distribution, for example the geometric distribution. Thus for each file, the server tosses a biased coin until it sees a tails and uses the number of heads (plus one) as the upload threshold. The problem here is the potential for infinite bandwidth overhead, so it makes sense to bound the threshold by the finite limit $\mathsf{B}$ and truncate the distribution at that point. However this means that, depending on parameter choices, we could get a high probability of $\mathsf{thr} = \mathsf{B}$ which could aid the adversary. Alternative distributions [2, 10, 17] have been proposed in the literature and we compare some of the other approaches later in Section 5.

# 3. MODELING DEDUPLICATION

This section presents a formal model of the existence-of-file attack on client-side deduplication by giving an indistinguishability-based notion of security. Client-side deduplication incorporates the following interaction between client and server, and the crucial item is the signal $\mathtt{sig}$ sent from the server to the client to indicate whether it wants the client to upload the file ($\mathtt{sig} = 1$) or not ($\mathtt{sig} = 0$).

1. When a client uploads a file, it will send a short description $h_F$ to the server (or otherwise enable the server to decide if it has the file).
2. Server will then communicate a response $\mathtt{sig}$ to the client.
3. Client then sends the file if required.

Note that the first and third steps may be preceded by encryption or segmentation, and thus we focus on file-based deduplication to simplify our results. As we mentioned earlier, this signal potentially gives the client the ability to learn whether or not a file is already stored.

Note that once the server gives $\mathtt{sig} = 0$ to some client, it should give the signal 0 to every subsequent upload request for that file. We make this assumption to strengthen our

adversary: in practice the adversary may not be sure that she has been the only person to upload a file in a given time period, but in our idealized model any requests for upload after the first instance of $\mathtt{sig} = 0$ will only result in wasted bandwidth.

## 3.1 Deduplication Strategies

Cloud providers that are concerned with the potential consequences of the existence-of-file attack may wish to implement a *deduplication strategy* that chooses the upload threshold based on some probability distrubution. This approach will ideally reduce an adversary's ability to gain information from its uploads, in a way that does not severely impact the amount of bandwidth required. In Section 4 we discuss the important tradeoff between the bandwidth overhead and the security gain.

To define client-side deduplication strategies for cloud storage systems, we regard strategies as distributions on the possible thresholds. By this we mean a strategy $\mathsf{DS}$ can be written as a list $(p_0 = 0, p_1, \dots)$ where $p_i$ is the probability that the threshold is value $i$. The first upload request must be met with the signal 1 (otherwise the file could not be retrieved) so $p_0 = 0$ for all meaningful strategies: we will subsequently forego writing $p_0$ when representing strategies. We refer to $\mathsf{DS}$ as the probability mass function for the strategy, and $\mathsf{DS.Alg}$ as the algorithm that implements strategy $\mathsf{DS}$. More formally:

Definition 1 (Deduplication Strategy). *A deduplication strategy* $\mathsf{DS}$ *is characterized by its probability distribution*

$$\mathsf{DS}(F, \lambda) = (p_1(F, \lambda), p_2(F, \lambda), \dots)$$

*where* $p_i(F, \lambda) = \mathbf{Pr}\left[i \leftarrow \mathsf{DS.Alg}(F, \lambda)\right]$. *A threshold selection algorithm* $\mathsf{DS.Alg}$ *is a probabilistic procedure that on input a deduplication strategy distribution* $\mathsf{DS}$*, security parameter* $\lambda \in \mathbb{N}$ *and a file* $F \in \{0,1\}^*$*, outputs a threshold* $\mathsf{thr} \in \mathbb{N}$*. Denote this event by* $\mathsf{thr} \leftarrow \mathsf{DS.Alg}(F, \lambda)$.

We say that a $\mathsf{DS}$ is *file-oblivious* if the distributions are independent of the file, i.e. $\mathsf{DS.Alg}(F, \lambda) = \mathsf{DS.Alg}(F^*, \lambda)$ for all $\lambda \in \mathbb{N}$ and all $F, F^* \in \{0,1\}^*$. Moreover, we say that $\mathsf{DS}$ is *finite* if for all security parameters $\lambda$ and files $F \in \{0,1\}^*$, there exists an upper bound $\mathsf{B} = \mathsf{B}(F, \lambda)$ such that $p_j(F, \lambda) = 0$ for all $j > \mathsf{B}(F, \lambda)$. While our model does not discount strategies that are file-dependent, we have not found any examples of such a strategy existing in the literature. Consequently, we only consider finite and file-oblivious strategies for the rest of the paper. For clarity, we omit the file and security parameter inputs and write $p_i$ instead of $p_i(F, \lambda)$.

Note that this definition includes strategies where, for example, the adversary flips a biased coin for each upload to decide if the server should stop requesting uploads – this strategy is included by flipping the coin until a threshold is given. Because we only consider finite strategies, this means that for any strategy that flips coins, the probability of the final threshold value $p_{\mathsf{B}}$ is forced to be $1 - \sum_{i=1}^{\mathsf{B}-1} p_i$. Some special cases include:

- Server that does not defend against existence-of-file attack $\mathsf{DS}^{\mathrm{dnd}} = (1, 0, 0, \dots)$
- Threshold chosen uniformly at random [7] $\mathsf{DS}^{\mathsf{U}} = (\frac{1}{\mathsf{B}}, \frac{1}{\mathsf{B}}, \dots, \frac{1}{\mathsf{B}}, 0, \dots)$
- Fair coin $\mathsf{DS}^{\mathrm{fc}} = (\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \dots, \frac{1}{2^{\mathsf{B}-1}}, 0, \dots)$

## 3.2 Security Model for Existence-of-File Attack

We now introduce the notion of indistinguishability under existence-of-file attack (IND-EFA) and give our security experiment for an adversary mounting an existence-of-file attack. The general IND-EFA experiment for deduplication schemes is depicted in Figure 1. At the start of the security experiment, $\mathcal{A}$ chooses a file $F$ from the filespace (denoted by $F \leftarrow \mathcal{A}$). Here the adversary $\mathcal{A}$ attempts to distinguish the two distributions $\mathsf{DS}$ and $\mathsf{DS}^*$, where $\mathsf{DS}^*$ is the deduplication strategy probability function shifted one position to the left. The experiment keeps a counter $\mathsf{ctr}$ to track how many upload requests have been performed for file $F$. The challenger, acting as the server, then invokes strategy algorithm $\mathsf{DS.Alg}$ on $F$ and in the $b = 1$ case increments the counter by one to simulate initial storage of $F$. $\mathcal{A}$ has access to a $\mathsf{store}()$ oracle which increments the storage counter and responds with the appropriate signal $\mathtt{sig}$. As a consequence of this, if the adversary makes more than $\mathsf{B}$ queries to the $\mathsf{store}()$ oracle, it will always receive $\mathtt{sig} = 0$ and thus not gain any information – this gives us an inherent bound on the number of (useful) queries an adversary can make. The experiment is parameterised by integer $\lambda$ that is an input to the $\mathsf{DS.Alg}$ algorithm. The line **return** $b' = b$ means that the experiment outputs 1 if the adversary has output $b' = b$ and thus won the game, and outputs 0 if $b' \neq b$.

Definition 2. *Indistinguishability under existence-of-file attack (*IND-EFA*). The advantage of an adversary* $\mathcal{A}$ *in the existence-of-file attack game against deduplication strategy* $\mathsf{DS}$ *is stated as follows:*

$$\mathbf{Adv}^{\mathsf{IND\text{-}EFA}}_{\mathsf{DS},\,\mathcal{A}}(\lambda) \stackrel{def}{=} \left| 2 \cdot \mathbf{Pr}\left[\mathbf{Exp}^{\mathsf{IND\text{-}EFA}}_{\mathsf{DS},\,\mathcal{A}}(\lambda) = 1\right] - 1 \right|$$

*where the experiment* $\mathbf{Exp}^{\mathsf{IND\text{-}EFA}}_{\mathsf{DS},\,\mathcal{A}}(\lambda)$ *is given in Figure 1.*

As we mentioned earlier, there are multiple side channel attack vectors in the context of cloud storage, and this security experiment considers the case when the adversary is attempting to learn the storage status of one particular file. A straightforward hybrid argument extends our model to one with multiple files, but note that this would not accurately model an adversary attempting to learn which file from a set is stored on a cloud server. To see this, observe that in a multi-file extension of Figure 1, the challenger either stores or does not store each file that the adversary queries, meaning that an adversarial win indicates that it can distinguish the scenario when all or none of a set of files are stored.

$\underline{\mathbf{Exp}^{\mathsf{IND\text{-}EFA}}_{\mathsf{DS.Alg},\,\mathcal{A}}(\lambda):}$
$b \stackrel{\$}{\leftarrow} \{0,1\}$
$F \leftarrow \mathcal{A}$
$\mathsf{thr} \leftarrow \mathsf{DS.Alg}(F, \lambda)$
$\mathsf{ctr} \leftarrow b$
$b' \leftarrow \mathcal{A}^{\mathsf{store}}(\lambda)$
**return** $b' = b$

$\mathsf{store}()$:
$\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
**if** $\mathsf{ctr} < \mathsf{thr}$ **then**
$\mathtt{sig} \leftarrow 1$
**else**
$\mathtt{sig} \leftarrow 0$
**return** $\mathtt{sig}$

**Figure 1: The general IND-EFA experiment for deduplication schemes.**

Note that Definition 2 is expressed in terms of *files* and implicitly assumes that the CSP uses file-based deduplication. We could equally express the definition in terms of *blocks* where the adversary is aiming to find whether a specific block has been uploaded, and if the storage protocol involves deterministic client-side segmentation then our model directly applies. However, we should be careful to note that if the server does use block-based deduplication, but the adversary is trying to test whether a specific file made up of different blocks is already uploaded, then a different notion would be needed. It is conceivable that an adversary could learn that all the blocks of its target file are stored, and mistakenly conclude that the file is stored. Additionally, a naive server handling unencrypted data may wish to only defend 'important blocks' (e.g. block in a contract that includes the salary field), but this in fact gives the adversary more power as she can then confirm that her contract template is correct. A full treatment of the block-based scenario sits outside of the scope of our model and we consider it future work.

One issue that is deliberately omitted from our analysis is the process used by the CSP to handle deletion. If the threshold is met for a specific file but subsequently all users delete the file from their storage then the CSP may be tempted to remove the file plus associated data, including $\mathsf{thr}$ from its storage. However, this means that the next time a client uploads that file the CSP would need to send $\mathtt{sig} = \mathtt{1}$ and randomly choose a new $\mathsf{thr}$. This gives rise to a subtle attack: an adversary creates $\mathsf{B}$ clients, uploads file $F$ $\mathsf{B}$ times, observes threshold $\mathsf{thr}_1$ then deletes all its instances of $F$. She then repeats this procedure by uploading the file $\mathsf{B}$ times and observes threshold $\mathsf{thr}_2$. If $F$ is stored by another user then $\mathsf{thr}_1 = \mathsf{thr}_2$ (i.e. actual threshold minus one), but if $F$ is not stored then $\mathsf{thr}_1 \neq \mathsf{thr}_2$. Thus this attack wins the IND-EFA game with probability almost 1 (with uncertainty only when $\mathsf{thr}_2$ is randomly chosen to be equal to $\mathsf{thr}_1$). For this reason, we suggest that file-storage counters should be non-decreasing and servers should not delete files once they are uploaded: we understand that the second criterion is widely deployed already by CSPs.

# 4. SECURITY AND EFFICIENCY TRADE-OFFS

In this section we analyze some desirable properties of effective deduplication strategies. We first show that non-increasing strategies optimize both efficiency (in terms of bandwidth costs) and security. Thus all good strategies should be non-increasing. Then we obtain a bound on the product of efficiency and security, characterizing an essential trade-off between bandwidth overhead and defence against the existence-of-file attack, for any deduplication strategy. As a corollary we prove that the uniform strategy proposed by Harnik et al. [7] is optimal, using the product of efficiency and security as a natural efficacy metric.

Let $\mathsf{DS} = (p_1, p_2, \dots)$ be a deduplication strategy. The expected bandwidth cost (in terms of the number of expected uploads of each file) can be quantified by computing the expected threshold

$$E = \sum_{i=1}^{\infty} i p_i.$$

This is a natural measure of the overhead cost of the deduplication strategy: the strategy that does not defend against the existence-of-file attack has $E = 1$. For finite strategies, this sum will always converge.

The security of a deduplication strategy is measured by the advantage defined in Section 3. The adversary only gains information when $p_i \neq p_{i+1}$. The scenario described earlier, where the adversary successfully launches the existence-of-file attack by being told not to upload on its first attempt and winning, corresponds to the difference between $p_0$ and $p_1$, which equals the probability $p_1$. Since the adversary's job is essentially to distinguish two probability distributions (the original distribution and its shift by one), the statistical distance of the two distributions

$$\Delta = \sum_{i=0}^{\infty} |p_i - p_{i+1}|$$

is an upper bound on the advantage in the IND-EFA game, and hence a useful security measure.

We now state our first theorem, which shows that *non-increasing strategies*, that is strategies $\mathsf{DS}' = (p'_1, p'_2, \dots, p'_{\mathsf{B}}, 0, \dots)$ where $p'_1 \geq p'_2 \geq \cdots \geq p'_{\mathsf{B}}$, minimize both the expected bandwidth cost $E$ and the security level $\Delta$. This result shows that the deduplication strategy of Lee and Choi [10] is sub-optimal under any reasonable security metric, meaning one that considers the probability that the adversary learns *some information* about a file's storage status (rather than only considering instances where the adversary is certain). We discuss their approach in more detail in Section 5.

THEOREM 1. *Let $\mathsf{DS} = (p_1, p_2, \dots, p_{\mathsf{B}}, 0, \dots)$ be any deduplication strategy, and let $\Delta$ and $E$ be the corresponding values. Let $\pi$ be a permutation on $\{1, 2, \dots, \mathsf{B}\}$ such that $\mathsf{DS}' = (p_{\pi(1)}, p_{\pi(2)}, \dots, p_{\pi(\mathsf{B})}, 0, \dots)$ is a non-increasing deduplication strategy with corresponding values $\Delta'$ and $E'$. Then $\Delta' \leq \Delta$ and $E' \leq E$.*

PROOF. We first show that $E' \leq E$. Suppose indexes $i, j$ exist with $i < j$ and $p_i < p_j$. Consider the strategy $\mathsf{DS}^* = (\dots, p_{i-1}, p_j, p_{i+1}, \dots, p_{j-1}, p_i, p_{j+1}, \dots)$. It is clear that expected value for the two strategies differ only in the two terms involving $p_i$ and $p_j$, respectively $ip_i + jp_j$ and $ip_j + jp_i$. If $p_j = p_i + \delta$, then

$$ip_i + jp_j = (i+j)p_i + \delta j > (i+j)p_i + \delta i = ip_j + jp_i.$$

It is then clear that by successive swaps such as this one, we can construct a permutation $\pi$ on $\{1, 2, \dots, \mathsf{B}\}$ such that $p_{\pi(1)} \geq p_{\pi(2)} \geq \cdots \geq p_{\pi(\mathsf{B})}$. Since none of these swaps increase the expectation, the first claim holds.

Next, we show that $\Delta' \leq \Delta$. Any deduplication strategy $\mathsf{DS}$ can be covered by one of the following cases:

1. There are indexes $0 < j < k < m < \mathsf{B}$ such that $p_{j-1} < p_j$, $p_k < p_j$, $p_k < p_m$, $p_{m+1} < p_m$ and $p_i$ is non-decreasing for $i = k, \dots, m$ and non-increasing for $i = j, \dots, k$ and $i = m, \dots, \mathsf{B}$.
2. There is an index $1 < k < \mathsf{B}$ such that $p_1 \leq p_2 \leq \cdots \leq p_k$ and $p_k \geq p_{k-1} \geq \cdots \geq p_{\mathsf{B}}$.
3. The strategy is non-increasing: $p_1 \geq p_2 \geq \cdots \geq p_{\mathsf{B}}$.
4. The strategy is non-decreasing: $p_1 \leq p_2 \leq \cdots \leq p_{\mathsf{B}}$.

In the first case, let $\pi$ be a permutation leaving $\{1, 2, \dots, j-1\}$ fixed and satisfying $p_{\pi(j)} \geq p_{\pi(j+1)} \geq \cdots \geq p_{\pi(\mathsf{B})}$, and let

$\mathsf{DS}'' = (p_{\pi(1)}, p_{\pi(2)}, \ldots, p_{\pi(\mathsf{B})}, 0, \ldots)$. If $p_j \geq p_m$, a calculation shows that

$$\Delta - \Delta'' = -p_k + p_m - p_k + p_m = 2(p_m - p_k) > 0.$$

Likewise, if $p_j < p_m$, another calculation shows that

$$\Delta - \Delta'' = (p_j - p_k) + (p_m - p_k) - (p_m - p_j) = 2(p_j - p_k) > 0.$$

It is clear that this operation can be applied repeatedly, eventually resulting in a strategy for which the first case does not apply, and for which the statistical distance is smaller than for $\mathsf{DS}$.

We have now shown that there is a sequence of permutations on $\{1, 2, \ldots, \mathsf{B}\}$ that can be composed into a permutation $\pi$ such that the strategy $\mathsf{DS}' = (p_{\pi(1)}, \ldots, p_{\pi(\mathsf{B})}, 0, \ldots)$ is non-increasing and has no greater statistical distance than $\mathsf{DS}$.

In the second case, we have $\Delta = p_k + p_k = 2p_k$. The non-increasing strategy $\mathsf{DS}'$ in the second case has $p_{\pi(1)} = p_k$. Then, we have $\Delta = \Delta' = 2p_k$. In the third case, it is trivial to show that $\Delta = \Delta' = 2p_1$. In the fourth case, we have $\Delta = \Delta' = 2p_B$. Then for any deduplication strategy, $\Delta' \leq \Delta$ holds.

□

We now present our second theorem, which describes a lower bound on the value of $\Delta E$ for any non-increasing deduplication strategy, in terms of the bound $B$. As may be expected, this shows that there must be a trade-off between the security level $\Delta$ and the efficiency $E$.

THEOREM 2. *Let $\mathsf{DS}' = (p'_1, p'_2, \ldots, p'_{\mathsf{B}}, 0, \ldots)$ be a non-increasing deduplication strategy, and let $\Delta'$ and $E'$ be its corresponding values. Then $1 + \frac{1}{\mathsf{B}} \leq \Delta' E'$.*

PROOF. First note that for the uniform deduplication strategy, $\Delta^{\mathsf{U}} E^{\mathsf{U}} = \frac{2}{\mathsf{B}} \cdot \frac{\mathsf{B}+1}{2} = \frac{\mathsf{B}+1}{\mathsf{B}} = 1 + \frac{1}{\mathsf{B}}$. Our proof strategy is to show that 'evening out' non-increasing strategies in a particular way so that they become 'more uniform' minimizes $\Delta \cdot E$. The idea here is to start at $p_1$ and move rightwards: For the first $j$ such that $p_j > p_{j+1}$, we set $p_1^* = \cdots = p_{j+1}^* = \frac{(j-1)p_1 + p_{j+1}}{j}$ and show that invoking this procedure does not increase the value of $\Delta \cdot E$. If we apply this procedure incrementally we will eventually have the uniform deduplication strategy $\mathsf{DS}^{\mathsf{U}} = (\frac{1}{\mathsf{B}}, \frac{1}{\mathsf{B}}, \ldots, \frac{1}{\mathsf{B}}, 0, \ldots)$. We start with a non-increasing strategy

$$\mathsf{DS}' = (p'_1, \ldots, p'_{\mathsf{B}}, 0, \ldots)$$
$$p'_i = p'_{i+1} + \delta_i \text{ where } \delta_i \in [0, 1],\ \forall i \in \{2, \ldots, \mathsf{B}\}.$$

Let $j$ be the first index such that $\delta_j \neq 0$ and define a new strategy as follows:

$$p_i^* = \begin{cases} p_1 - \frac{\delta_j}{j} & \text{if } i \in \{1, \ldots, j\}, \\ p_{j+1} + \frac{(j-1)\delta_j}{j} & \text{if } i = j+1, \\ p_i & \text{otherwise.} \end{cases}$$

This process gives us a new deduplication strategy

$$\mathsf{DS}^* = (p_1^*, \ldots, p_{j+1}^*, p'_{j+2}, \ldots, p'_{\mathsf{B}}, 0, \ldots).$$

We repeat this process until we have the uniform distribution, and will now show that each step does not increase $\Delta \cdot E$. Denote $\Delta_{\mathtt{old}}$ and $\Delta_{\mathtt{new}}$ as the statistical distance before and after each iteration of the process described above respectively. We can calculate $\Delta_{\mathtt{new}}$ as follows. Observe that for any non-increasing strategy, $\Delta = 2 \cdot p_1$. Then for invoking the above process at position $r$,

$$\Delta_{\mathtt{new}} = 2 \cdot p_1^* = 2 \cdot (p_1 - \frac{\delta_r}{r}) = \Delta_{\mathtt{old}} - \frac{2\delta_r}{r}.$$

Similarly for the expected threshold,

$$\begin{aligned} E_{\mathtt{new}} &= \sum_{i=1}^{\mathsf{B}} i \cdot p_i^* \\ &= p_1^* + \cdots + r \cdot p_r^* + (r+1) \cdot p_{r+1}^* + \sum_{i=r+2}^{\mathsf{B}} i \cdot p_i^* \\ &= (p_1 - \frac{\delta_r}{r}) + \cdots + r \cdot (p_r - \frac{\delta_r}{r}) \\ &\quad + (r+1) \cdot (p_{r+1} + \frac{(r-1)\delta_r}{r}) + \sum_{i=r+2}^{\mathsf{B}} i \cdot p_i^* \\ &= \sum_{i=1}^{\mathsf{B}} i \cdot p_i - \frac{\delta_r}{r}(1 + 2 + \cdots + r) + \frac{\delta_r}{r}(r+1)(r-1) \\ &= E_{\mathtt{old}} + \frac{\delta_r}{r}\left[ - \frac{r(r+1)}{2} + (r-1)(r+1) \right] \\ &= E_{\mathtt{old}} + \frac{(r+1)(r-2)\delta_r}{2r}. \end{aligned}$$

This means that

$$\begin{aligned} \Delta_{\mathtt{new}} \cdot E_{\mathtt{new}} &= \left[ \Delta_{\mathtt{old}} - \frac{2\delta_r}{r} \right] \cdot \left[ E_{\mathtt{old}} + \frac{(r+1)(r-2)\delta_r}{2r} \right] \\ &= \Delta_{\mathtt{old}} \cdot E_{\mathtt{old}} + \frac{(r+1)(r-2)\delta_r}{2r} \cdot \Delta_{\mathtt{old}} \\ &\quad - \frac{2\delta_r}{r} \cdot E_{\mathtt{old}} - \frac{\delta_r^2}{r^2}(r+1)(r-2) \end{aligned}$$

and thus if we want to show that the process minimizes the value of $\Delta \cdot E$ then we need to show that $\Delta_{\mathtt{new}} \cdot E_{\mathtt{new}} - \Delta_{\mathtt{old}} \cdot E_{\mathtt{old}} \leq 0$, or equivalently

$$\frac{(r+1)(r-2)\delta_r}{2r} \cdot \Delta_{\mathtt{old}} - \frac{2\delta_r}{r} \cdot E_{\mathtt{old}} - \frac{(r+1)(r-2)\delta_r^2}{r^2} \leq 0$$

To show this, note that since $\delta_r \in (0, 1]$ and $r \in \mathbb{N}_+$ we can multiply by $\frac{2r^2}{\delta_r}$:

$$r(r+1)(r-2)\Delta_{\mathtt{old}} - 4rE_{\mathtt{old}} - 2(r+1)(r-2)\delta_r \leq 0. \quad (1)$$

Now we use the fact that $\Delta_{\mathtt{old}} = 2 \cdot p_1$ and

$$\begin{aligned} E_{\mathtt{old}} &= \sum_{i=1}^{\mathsf{B}} i p_i \\ &= p_1 + \sum_{i=2}^{r} i p_1 + (r+1) p_{r+1} + \sum_{i=r+2}^{\mathsf{B}} i p_i \\ &= p_1 \cdot \frac{r(r+1)}{2} + (r+1)(p_1 - \delta_r) + \sum_{i=r+2}^{\mathsf{B}} i p_i \\ &= \frac{(r+1)(r+2)}{2} p_1 - \delta_r - r\delta_r + \sum_{i=r+2}^{\mathsf{B}} i p_i \end{aligned}$$

and plug this into equation 1 we get

$$2r(r-1)p_2 - 2rp_2(r+3) + 4\delta_r + 4r\delta_r - 4\sum_{i=r+2}^{\mathsf{B}} ip_i - 2\delta_r(r-1) \leq 0$$

$$\Leftrightarrow \quad -8rp_2 + 2\delta_r + 2r\delta_r - 4r\sum_{i=r+2}^{\mathsf{B}} ip_i \leq 0.$$

Since the summation term is non-negative and $(r+1) > 0$, we need to show that

$$4rp_1 \geq (r+2)\delta_r$$

We know that the strategy is non-increasing so $p_1 \geq \delta_r$, and since $r \in \{1, \ldots, \mathsf{B}\}$ we are sure that $4r \geq (r+2)$ so we are done.

If we repeat this process until we reach $p'_{\mathsf{B}-1}$ then we will have the uniform strategy $\mathsf{DS}^{\mathsf{U}}$ as required. □

We now claim that $\Delta \cdot E$ is a natural efficacy metric for deduplication strategies, giving equal weight to bandwidth efficiency and security against the existence-of-file attack, where the former is measured by the expected bandwidth cost $E$ and the latter is measured by the upper bound $\Delta$ on the adversary's advantage. We remark that other metrics are possible and may even be more appropriate for certain circumstances, and it is plausible that similar results could be obtained for other metrics. We now bring together Theorem 1 and Theorem 2 and give the following corollary, which states that the uniform strategy (i.e. choosing the threshold uniformly at random) is optimal in terms of this particular metric.

COROLLARY 1. *Let* $\mathsf{DS}^{\mathsf{U}} = (\frac{1}{\mathsf{B}}, \frac{1}{\mathsf{B}}, \ldots, \frac{1}{\mathsf{B}}, 0, \ldots)$ *be the uniform deduplication strategy with corresponding values* $E^{\mathsf{U}}$ *and* $\Delta^{\mathsf{U}}$. *Then for any deduplication strategy* $\mathsf{DS} = (p_1, p_2, \ldots, p_{\mathsf{B}}, 0, \ldots)$ *with corresponding values* $E$ *and* $\Delta$,

$$E^{\mathsf{U}}\Delta^{\mathsf{U}} \leq E\Delta.$$

# 5. RELATED WORK

Lee and Choi [10] suggest using a variable threshold by making a random choice at each upload, but note that this is equivalent to making all the random choices at the start and simply induces a probability distribution on the threshold as in our model. Lee and Choi claim that their solution provides better security than Harnik et al.'s uniform random choice [7] while having the same efficiency (expected number of uploads is $\frac{\mathsf{B}+1}{2}$). However, their measure of security basically states that the adversary wins only if she is certain that the file was uploaded or not. This is rather like requiring message recovery security for encryption rather than the more usual and stronger indistinguishability requirement. The security measure that we consider in this paper only requires the adversary to learn whether or not the file was uploaded with significant probability. In this stronger security model, the scheme of Lee and Choi is in fact weaker than Harnik et al.'s uniform choice of probability.

Shin and Kim [17] discuss *related-files* attack and claim that both Harnik et al.'s protocol [7] and Lee and Choi's protocol [10] are vulnerable to this attack because those schemes assume that all files are stored independently. By related-files attack, Shin and Kim mean situations where files are correlated to each other and thus stored at the same server together, e.g. files on a software package or document files with the same content in different file formats such as doc, pdf and xml. The adversary then tries to identify whether any one of a number of related files have been uploaded. By uploading not only $F$ but also other files that are related to $F$, the adversary may infer the existence of $F$ with higher probability than the case of independent files. In this scenario the randomization would have to be increased in order to maintain the same level of security. However, their proposed solution includes using a trusted storage gateway at the edge of the client's network, which in general seems impractical to assume. We do not consider this sort of attack in our model.

Wang et al. [19] adopt a game-theoretic approach to model the side-channel attacks in cross-user client-side deduplication. They consider the scenario as a non-cooperative game played between an adversary and the CSP, which means the players are assumed to share no information with others during a game. The game is assumed to be dynamic which means that the game is played more than once (as the players have to learn their opponents' payoff through repeated game iterations). The players will focus on optimizing their own payoff. Wang et al. claim that their proposed solution requires significantly fewer uploads than those required by Harnik et al.'s randomized threshold-based solution [7], and conclude that they achieve improved efficiency in terms of reduced bandwidth overhead. However, their results are based on comparisons using their chosen payoff matrix which puts a specific value on a successful attack as well as the cost of uploads. Our results instead give a fixed bound on how security and efficiency interact and we can increase efficiency arbitrarily by reducing security level in compensation. In practice, economic aspects will influence the chosen trade-off between security and efficiency.

Ács et al. [2] examine *cache privacy attacks* on Named Data Networking. Although this is a completely different application area from deduplication in the cloud, they propose a strongly related mechanism designed to prevent the adversary from learning whether a particular data item has been stored in a local cache. They propose techniques to "randomly decide whether to mimic a cache hit or a cache miss". They compare privacy and efficiency, and use $(\epsilon, \delta)$-probabilistic indistinguishability as the privacy measure, somewhat similar to $\Delta$ in our model. Their measure of efficiency, which they call *utility*, relates the number of cache hits with the total number of file requests. This is in contrast to our measure of efficiency where we do not consider the overall popularity of a file. The Ács et al. [2] model has some details which our model ignores; however they cannot express the security/efficiency trade-off in a manner as concise as our results. We expect that there can be some benefit in trying to combine the advantages of both approaches, in particular relating to the multiple-file IND-EFA extension described in Section 3.2.

Another similar scenario for data deduplication is *memory deduplication*. In memory deduplication, memory pages with the same contents are merged which reduces the memory footprint of a running system. Memory deduplication has applications both in virtualization solutions (to host more virtual machines with the same amount of physical memory) and operating systems. It has been adopted as a default feature in Windows 8.1 (and later versions) [4]. Kernel Same-

page Merging (KSM) is the Linux implementation of memory deduplication that uses a kernel thread for periodically scanning memory and finding memory pages with the same contents which should be merged [15]. However, memory deduplication provides side-channel information and causes security problems. The side-channel information incurred by memory deduplication in virtualized environments has been exploited in recent work [4,15] and again we expect our model to be of some utility in creating solutions that defend against these attacks: if the KSM module were to use randomized thresholds for deduplication of memory pages then the tradeoff between efficiency and security is very similar to the cloud storage scenario covered in this paper.

# 6. CONCLUDING REMARKS

A secure client-side deduplication scheme should defend against side-channel attacks whereby an attacker attempts to determine whether or not a specific file exists on the cloud. In this paper we showed how to model such attacks and analyzed solutions based on probabilistic uploads. We gave conditions on the strategies that servers should employ when defending against these attacks. We then showed that the uniform distribution for probabilistic uploads provides the optimal solution for a natural measure, which presents a tradeoff between security and bandwidth usage.

# 7. ACKNOWLEDGEMENTS

We would like to thank Håvard Raddum for helpful discussions and the anonymous reviewers for their feedback. This research was funded by The Research Council of Norway under Project No. 248166.

# 8. REFERENCES

[1] M. Abadi, D. Boneh, I. Mironov, A. Raghunathan, and G. Segev. Message-locked encryption for lock-dependent messages. In *Advances in Cryptology – CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 374–391, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[2] G. Ács, M. Conti, P. Gasti, C. Ghali, and G. Tsudik. Cache privacy in Named-Data Networking. In *IEEE 33rd International Conference on Distributed Computing Systems, ICDCS 2013*, pages 41–51. IEEE Computer Society, 2013.

[3] M. Bellare, S. Keelveedhi, and T. Ristenpart. Message-locked encryption and secure deduplication. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece*, volume 7881 of *Lecture Notes in Computer Science*, pages 296–312. Springer, 2013.

[4] E. Bosman, K. Razavi, H. Bos, and C. Giuffrida. Dedup Est Machina: memory deduplication as an advanced exploitation vector. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 987–1004, May 2016.

[5] 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 *ICDCS*, pages 617–624, 2002.

[6] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg. Proofs of ownership in remote storage systems. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pages 491–500, New York, NY, USA, 2011. ACM.

[7] D. Harnik, B. Pinkas, and A. Shulman-Peleg. Side channels in cloud services: Deduplication in cloud storage. *IEEE Security & Privacy*, 8(6):40–47, 2010.

[8] T. Jiang, X. Chen, Q. Wu, J. Ma, W. Susilo, and W. Lou. Towards efficient fully randomized message-locked encryption. In *Proceedings of 21st Australasian Conference on Information Security and Privacy (ACISP 2016), Melbourne, Australia, July 4-6, 2016*, pages 361–375, Cham, 2016. Springer International Publishing.

[9] S. Keelveedhi, M. Bellare, and T. Ristenpart. Dupless: Server-aided encryption for deduplicated storage. In *Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA, August 14-16, 2013*, pages 179–194. USENIX Association, 2013.

[10] S. Lee and D. Choi. Privacy-preserving cross-user source-based data deduplication in cloud storage. In *2012 International Conference on ICT Convergence (ICTC)*, pages 329–330, Oct 2012.

[11] J. Liu, N. Asokan, and B. Pinkas. Secure deduplication of encrypted data without additional independent servers. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 874–885. ACM, 2015.

[12] D. T. Meyer and W. J. Bolosky. A study of practical deduplication. In G. R. Ganger and J. Wilkes, editors, *9th USENIX Conference on File and Storage Technologies*, pages 1–13. USENIX, 2011.

[13] M. Mulazzani, S. Schrittwieser, M. Leithner, M. Huber, and E. Weippl. Dark clouds on the horizon: Using cloud storage as attack vector and online slack space. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11, pages 5–5, Berkeley, CA, USA, 2011. USENIX Association.

[14] V. Rabotka and M. Mannan. An evaluation of recent secure deduplication proposals. *Journal of Information Security and Applications*, 27-28:3 – 18, 2016. Special Issues on Security and Privacy in Cloud Computing.

[15] K. Razavi, B. Gras, E. Bosman, B. Preneel, C. Giuffrida, and H. Bos. Flip feng shui: Hammering a needle in the software stack. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 1–18, Austin, TX, Aug. 2016. USENIX Association.

[16] H. Ritzdorf, G. O. Karame, C. Soriente, and S. Capkun. On Information Leakage in Deduplicated Storage Systems. In *Proceedings of the 8th Edition of the ACM Workshop on Cloud Computing Security*, CCSW '16. ACM, 2016.

[17] Y. Shin and K. Kim. Differentially private client-side data deduplication protocol for cloud storage services. *Security and Communication Networks*, 8(12):2114–2123, 2015.

[18] J. Stanek, A. Sorniotti, E. Androulaki, and L. Kencl. A secure data deduplication scheme for cloud storage. In N. Christin and R. Safavi-Naini, editors, *Financial*

*Cryptography and Data Security - 18th International Conference, FC 2014, Christ Church, Barbados*, volume 8437 of *Lecture Notes in Computer Science*, pages 99–118. Springer, 2014.

[19] B. Wang, W. Lou, and Y. T. Hou. Modeling the side-channel attacks in data deduplication with game theory. In *2015 IEEE Conference on Communications and Network Security (CNS)*, pages 200–208, Sept 2015.