Definitions for Plaintext-Existence Hiding in Cloud Storage

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

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

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

Håvard Raddum
Simula@UiB
Bergen, Norway
haavardr@simula.no

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

ABSTRACT

Cloud storage services use deduplication for saving bandwidth and storage. An adversary can exploit side-channel information in several attack scenarios when deduplication takes place at the client side, leaking information on whether a specific plaintext exists in the cloud storage. Generalising existing security definitions, we introduce formal security games for a number of possible adversaries in this domain, and show that games representing all natural adversarial behaviors are in fact equivalent. These results allow users and practitioners alike to accurately assess the vulnerability of deployed systems to this real-world concern.

CCS CONCEPTS

• **Security and privacy**; • **Information systems** → *Cloud based storage*;

KEYWORDS

Cloud Storage, Side-channel analysis, Information Leakage

1 INTRODUCTION

Outsourced storage is by now strikingly prevalent for individuals and enterprises. This booming industry has encouraged fierce competition between cloud storage providers (CSPs) to acquire new clients, and to ensure that existing customers do not move on. The competitive market has led to very low prices for vast amounts of storage, with some CSPs even giving free storage to new customers. There are two mechanisms that have allowed the CSPs to provide such startling prices and offers targeted advertising and (cross-user) deduplication. We note in passing that the first mechanism is often misunderstood by users: the data that they upload has inherent value making it worthwhile to leak to advertisers.

ARES 2018, August 27–30, 2018, Hamburg, Germany

ACM ISBN 978-1-4503-6448-5/18/08…$15.00
https://doi.org/10.1145/3230833.3234515

Deduplication is the process by which CSPs only store one copy of each file, irrespective of how many times that file is uploaded. If the CSP does this on the basis of individual clients then it will save storage in its data centers every time a client uploads a backup of a file. However the real storage savings can be made if the CSP enforces this procedure between users (cross-user deduplication): if Alice uploads a popular movie file and Bob subsequently wants to upload the same movie, the server only stores one copy and attaches a pointer to both Alice and Bob. In addition to a storage saving, the CSP can also save bandwidth by insisting that deduplication is done at the client side. Fig. 1 depicts the data flow in so-called *client-side deduplication*: a client initially sends a short descriptor (hash) of the file, the server then checks its storage and signals the client to upload the file only if the server does not have the hash value stored already.

The introduction of cross-user deduplication is good news for CSPs since they can pass on their savings to users in the form of lower prices. However, there is a potential conflict between deduplication and security since a user who encrypts files before uploading them destroys the server's ability to identify identical files. Several attempts have been made to provide *secure deduplication* [2, 3, 6–8, 10, 13–15] which allow client-side encryption without preventing deduplication. Such schemes have typically used deterministic encryption and message-dependent keys so that all encryptions of the same file are identical. This leads to degraded cryptographic security, and does not negate the following side channel that is present in *all* client-side deduplicating storage systems.

If the server informs the client that it does not require transmission of the full file then this (inherently) informs the client that

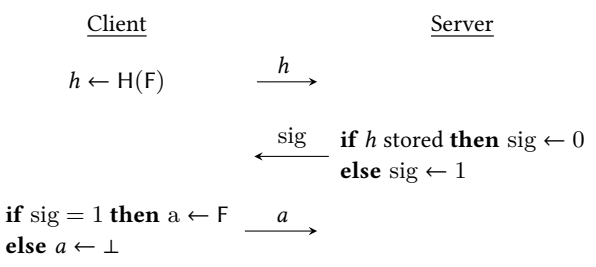

Figure 1: Information flow in client-side deduplication.

at least one copy of the file has already been stored by the server. This side channel has been studied in the literature [1, 4, 5, 11] and defending against it is intuitively straightforward: instead of producing $\mathrm{sig} = 1$ (please send full file) when a file is first uploaded and $\mathrm{sig} = 0$ (i.e. file already stored, no need to upload) for each subsequent upload, the server can return a noisy value that selects the signal according to some distribution for uploads (clearly the first upload necessitates $\mathrm{sig} = 1$). This creates a clear tradeoff between security (probability that an adversarial client can infer storage status of some file) and efficiency (number of expected uploads of each file).

The security–efficiency tradeoff was observed by Armknecht et al. [1], who also presented an indistinguishability-based security experiment to reflect adversarial capability in this attack scenario. However their model was very limited: it only considered an adversary that wished to learn whether a file was stored on the server or not – the archetypal example involves an employment contract of a person who uses the same CSP as the attacker. In many situations this analysis may not be sufficient. Consider that instead of trying to learn whether a contract is stored or not, the attacker Eve *knows* that Alice has stored her contract with the CSP and additionally knows many of the details (formatting, common text etc.) of such contracts, and creates a template file for this low-entropy document. Eve then performs the attack on each of these candidates, and if she receives $\mathrm{sig} = 0$ for any of them then she knows that she has found a match with Alice's real contract. This leaves a gap since it is not *a priori* clear whether an optimal solution for one security experiment is the same as for other natural experiments.

The goal of this paper is to fill this gap by providing a general security experiment for this attack vector. Further, we show that our definition is in fact equivalent to the one posited by Armknecht et al. [1], answering an open question from their work. This rigor will allow practitioners to accurately judge the risk to their systems posed by these real-world attack vectors, which subsequently allows users to make an informed decision about the security threats posed to their files. Continuing in this vein, we consider a generic security experiment for the effect of this class of attacks on fully-fledged cloud storage protocols. Such an approach has been missing from the literature so far, and this framework could potentially allow modular and natural analysis of the threat landscape in outsourced storage.

2 NOTATION

In this paper $x \leftarrow F(y)$ means that we assign x the output of procedure F on input of y. The code $x \xleftarrow{\$} S$ indicates that either x has been chosen uniformly from set S, or according to some distribution S. For vector $\mathbf{x}$, denote the number of components of $\mathbf{x}$ by $|\mathbf{x}|$, and the i^{th} component of $\mathbf{x}$ by $\mathbf{x}[i]$. We refer to *users* to mean entities that own an account with a CSP, and *clients* that represent the devices owned by users.

Our definitions of security follow the cryptographic literature by using a formal game that is played between a challenger and an idealized adversary. This adversary has access to some oracles and attempts to perform some task that defines success. This allows analysis of primitives and protocols via reductions to existing definitions or hard problems. If a reduction calls one of its own oracles then we indicate this by underlining the call. Pseudocode **return** $b' \stackrel{?}{=} b$ is used as shorthand for **if** $b' = b$ **then return** 1 // **else return** 0, with an output of 1 indicating adversarial success. We choose to follow the concrete security framework: an adversary's advantage is defined in terms of some generic security experiment and a protocol or primitive's security is then defined in terms of comparative statements and reductions. This removes a need for security parameters and a definition of a negligible-advantage adversary: a decision of particular utility in the cloud setting due to the vast computational capabilities of CSPs. If an adversary's probability of success is worse than guessing then we assume an automatic normalization to an adversary that is better than by guessing (this is possible since all of our adversaries output either a bit or an integer value). This assumption removes the requirement for absolute values in advantage statements.

3 PLAINTEXT HIDING IN DEDUPLICATING STORAGE SYSTEMS

Client-side deduplication provides savings in both storage and bandwidth. To upload a file, the client first uploads a short descriptor (usually a hash) of the file and then receives a signal that informs the client whether or not to go ahead with the full file upload. These savings come at a clear cost: when the server informs a client that it does not require a file to be uploaded then that client learns that the file is certainly already stored on the server. Most storage services that employ client-side deduplication will do the most simple and efficient solution: after a file has been uploaded once, tell all subsequent uploaders not to upload. If the server wishes to keep the storage status of a file hidden from other users then it must use some deduplication threshold selection strategy to choose how many times to ask for that file to be uploaded before informing clients that uploading the full file is not needed.

In *secure deduplication* schemes [2, 3, 6–8, 10, 14, 15], this 'file' is a ciphertext. These schemes attempt to hide the existence of the underlying plaintext on the storage from both clients and servers. However, in the client-side deduplication even with uploading the ciphertext, the existence of the corresponding plaintext can de easily deduced from the signal that the client receives from the server which indicates whether or not to upload data.

Harnik et al. (HPS) [4] (and subsequently Mulazzani et al. [11], Pulls [12] and Hovhannisyan et al. [5]) discussed the implications of this side channel in terms of three attacks:

(1) *Identifying files:* to identify storage of an incriminating file on the cloud, and possibly identifying its owner later with the help of law enforcement access.
(2) *Learning file contents:* to guess the contents of a file and infer its existence in the cloud.
(3) *Covert channels:* to use the existence or non-existence of a specific file on the cloud as a covert communication channel.

HPS noted that the mechanism for performing all of the above attacks is the same, and this was modelled more formally by Armknecht et al. [1] (henceforth ABDGT). However ABDGT's nomenclature – indistinguishability under existence-of-file attack (IND-EFA) – is imprecise: security definitions are normally expressed as {goal}-{cap}

for some property goal that a satisfying scheme will possess (e.g. indistinguishability of ciphertexts) and some capability cap that we suppose for an adversary (e.g. a chosen plaintext oracle). Thus we propose this notion should be called Plaintext-Existence Hiding under Chosen Store-Signal Attack (PEH-CSSA).

ABDGT considered a security experiment in which the adversary submits one file to a challenger, the challenger then flips a coin and either stores the file or not. The adversary then has access to an oracle that when fed a file, returns the (binary) deduplication signal. This game directly models the second HPS attack: Eve suspects that Alice, who also uses the same cloud storage provider, has stored her employment contract with the CSP. She then uploads the suspected contract and uses the deduplication signal to infer the storage status of Alice's contract. Any storage protocol that does not try to hide the storage status will not meet this notion of security. ABDGT went on to show that randomly choosing the deduplication threshold (the number of times a file is to be uploaded before the server tells clients not to send the file) balances security and efficiency.

An arguably more natural model for attacks aimed at learning file contents considers a dishonest client that knows that one of a set N of files is stored on the server. The client attempts to glean which one is stored (obvious example: Eve attempts to learn salary of Alice knowing that her salary can be one of finitely many values). ABDGT acknowledge this shortcoming and describe how a hybrid argument does not extend to this extended case, since the experiment would either store all N files or none. In real deployments the size of the set of possible files, the value N, could in fact be small (e.g. a discrete pay scale).

3.1 Plaintext Hiding Security Experiments

In the rest of this section we follow HPS and ABDGT and focus solely on the threshold selection algorithm DS.Alg, defined as in ABDGT:

Definition 1 (Deduplication Strategy). *A deduplication strategy* DS *is characterized by its probability distribution*

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

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

As in ABDGT, we note that this definition allows any strategy that depends in some way on the file being input, however we are not aware of any such schemes in the literature or in deployed systems – and in fact such an approach could potentially lead to further side-channel leakage. The most bandwidth-efficient solution – asking for the file once and informing subsequent uploaders not to send the full file – is represented by strategy $(1, 0, 0, \dots)$.

We will show how the '1 out of N' model is in fact implied by the decisional model considered by ABDGT. In Fig. 2 we detail a security experiment that we call $\mathsf{PEH_{dN}}$, which is a generalization of ABDGT's presentation (their results concern the $\mathsf{N} = 1$ case of $\mathsf{PEH_{dN}}$). In this experiment an adversary selects $\mathsf{N} \geq 1$ files, the challenger chooses one of the files and flips a coin to determine whether to store that file or not (d alludes to the 'decisional' nature of this definition).

Definition 2 ($\mathsf{PEH_{dN}}$-CSSA Security for Threshold Selection Strategies). *Let* DS.Alg *be a deduplication threshold selection strategy. Then the* $\mathsf{PEH_{dN}}$-CSSA *advantage for an adversary* $\mathcal{A}$ *against* DS.Alg *is defined by*

$$\mathbf{Adv}^{\mathsf{PEH_{dN}\text{-}CSSA}}_{\mathsf{DS.Alg},\,\mathcal{A}} = 2 \cdot \left[\mathbf{Pr}\left[\mathbf{Exp}^{\mathsf{PEH_{dN}\text{-}CSSA}}_{\mathsf{DS.Alg},\,\mathcal{A}} = 1\right] - \frac{1}{2}\right]$$

where experiment $\mathbf{Exp}^{\mathsf{PEH_{dN}\text{-}CSSA}}_{\mathsf{DS.Alg},\,\mathcal{A}}$ *is given in Fig. 2.*

$\mathbf{Exp}^{\mathsf{PEH_{dN}\text{-}CSSA}}_{\mathsf{DS.Alg},\,\mathcal{A}}$:

$b \xleftarrow{\$} \{0,1\}$
$i \xleftarrow{\$} \{1, \dots, \mathsf{N}\}$
$\mathbf{F} \leftarrow \mathcal{A}$
for $j \in \{1, \dots, \mathsf{N}\}$ **do**
  $\mathsf{thr}_{\mathbf{F}[j]} \leftarrow \mathsf{DS.Alg}(\mathbf{F}[j])$
for $j \in \{1, \dots, \mathsf{N}\} \setminus i$ **do**
  $\mathsf{ctr}_{\mathbf{F}[j]} \leftarrow 0$
$\mathsf{ctr}_{\mathbf{F}[i]} \leftarrow b$
$b' \leftarrow \mathcal{A}^{O.\mathsf{PEH.Store}}$
return $b' \stackrel{?}{=} b$

O.PEH.Store(j):
  $\mathsf{ctr}_{\mathbf{F}[j]} \leftarrow \mathsf{ctr}_{\mathbf{F}[j]} + 1$
  if $\mathsf{ctr}_{\mathbf{F}[j]} < \mathsf{thr}_{\mathbf{F}[j]}$ **then**
    $\mathsf{sig} \leftarrow 1$
  else
    $\mathsf{sig} \leftarrow 0$
  return sig

Figure 2: Plaintext-existence hiding experiments for deduplication schemes for yes/no variant $\mathsf{PEH_{dN}}$ (the special subcase $\mathsf{N} = 1$ is the original ABDGT experiment, denoted $\mathsf{PEH_d}$).

In Fig. 3 we detail a novel plaintext-existence hiding experiment, which we denote $\mathsf{PEH_N}$, where the adversary submits a vector of $\mathsf{N} \geq 2$ files, the challenger chooses one of them and the adversary aims to work out which one was stored. This is the natural extension discussed by ABDGT. Note the special subcase $\mathsf{N} = 2$, where the adversary submits just two files, is a standard indistinguishability-style game and we use a special label $\mathsf{PEH_{ind}}$ for this notion.

Definition 3 ($\mathsf{PEH_N}$-CSSA Security for Threshold Selection Strategies). *Let* DS.Alg *be a deduplication threshold selection strategy. Then the* $\mathsf{PEH_N}$-CSSA *advantage for an adversary* $\mathcal{A}$ *against* DS.Alg *is defined by*

$$\mathbf{Adv}^{\mathsf{PEH_N\text{-}CSSA}}_{\mathsf{DS.Alg},\,\mathcal{A}} = \frac{\mathsf{N}}{\mathsf{N}-1} \cdot \left[\mathbf{Pr}\left[\mathbf{Exp}^{\mathsf{PEH_N\text{-}CSSA}}_{\mathsf{DS.Alg},\,\mathcal{A}} = 1\right] - \frac{1}{\mathsf{N}}\right]$$

where experiment $\mathbf{Exp}^{\mathsf{PEH_N\text{-}CSSA}}_{\mathsf{DS.Alg},\,\mathcal{A}}$ *is given in Fig. 3.*

We emphasize that the O.PEH.Store oracle that the adversary has access to in these experiments takes a file pointer as input and returns the deduplication signal sig for the associated file, with $\mathsf{sig} = 1$ indicating that the client should upload the whole file and $\mathsf{sig} = 0$ indicating that transferring the file is not required.

Since we use concrete security throughout this paper, we must insist that all adversaries terminate after some 'reasonable' amount of time has passed to ensure that the reductions in the remainder

$\mathbf{Exp}^{\mathsf{PEH_N\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{A}}$:

$i \xleftarrow{\$} \{1, \ldots, \mathsf{N}\}$
$\mathbf{F} \leftarrow \mathcal{A}$
for $j \in \{1, \ldots, \mathsf{N}\}$ **do**
  $\mathsf{thr}_{\mathbf{F}[j]} \leftarrow \mathsf{DS.Alg}(\mathbf{F}[j])$
for $j \in \{1, \ldots, \mathsf{N}\} \setminus i$ **do**
  $\mathsf{ctr}_{\mathbf{F}[j]} \leftarrow 0$
$\mathsf{ctr}_{\mathbf{F}[i]} \leftarrow 1$
$i' \leftarrow \mathcal{A}^{O.\mathsf{PEH.Store}}$
return $i' \stackrel{?}{=} i$

O.PEH.Store(j):
$\mathsf{ctr}_{\mathbf{F}[j]} \leftarrow \mathsf{ctr}_{\mathbf{F}[j]} + 1$
if $\mathsf{ctr}_{\mathbf{F}[j]} < \mathsf{thr}_{\mathbf{F}[j]}$ **then**
  $\mathrm{sig} \leftarrow 1$
else
  $\mathrm{sig} \leftarrow 0$
return sig

Figure 3: Plaintext-existence hiding experiments for deduplication schemes for 1-of-N variant $\mathsf{PEH_N}$ (note special subcase $\mathsf{N} = 2$, denoted $\mathsf{PEH_{ind}}$).

of this section work. Each reduction is otherwise straightforward, and we use figures to clearly indicate how each reduction responds to the underlying adversary's queries.

3.2 Relations between Notions for Plaintext-Existence Hiding

In Fig. 4 we depict relations between notions: arrows denote reductions via the corresponding theorems that are presented in this section. The arrow for $\mathsf{PEH_N}$ implies $\mathsf{PEH_d}$ (Thm. 4) is starred because this reduction is not tight. These games only consider the deduplication threshold strategy algorithm as a standalone object, ignoring the wider protocol in which it exists. This is notable because this result appears, intuitively at least, to not necessarily be correct: schemes that protect which of a group of N files have been stored do not necessarily hide whether one file has been stored or not since an adversary can simply eavesdrop on the communication between the client and the server. This observation is further motivation for expanding the discussion of PEH attacks in wider protocols, as discussed in Section 4. Our first theorem shows the converse statement, that $\mathsf{PEH_d}$ implies $\mathsf{PEH_N}$.

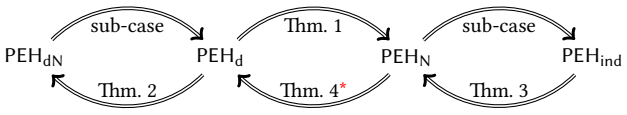

Figure 4: Relations between notions for plaintext-existence hiding in cloud storage.

Theorem 1. *Let* DS.Alg *be a deduplication threshold selection strategy. For any adversary $\mathcal{A}_1$ against $\mathsf{PEH_N}$ then there exists an adversary $\mathcal{B}_1$ of comparable computational complexity against $\mathsf{PEH_d}$ such that:*

$$\mathbf{Adv}^{\mathsf{PEH_N\text{-}CSSA}}_{\mathcal{A}_1} = \frac{\mathsf{N}}{\mathsf{N}-1}\mathbf{Adv}^{\mathsf{PEH_d\text{-}CSSA}}_{\mathcal{B}_1}.$$

Proof. The reduction is detailed in Fig. 5. When $\mathcal{B}_1$ receives the vector of files from $\mathcal{A}_1$ it needs to select one of them and use that in its own game, then when $\mathcal{A}_1$ calls its O.PEH.Store oracle on all other files $\mathcal{B}_1$ needs to simulate those calls as if the files had not been stored by $\mathcal{A}_1$'s challenger.

If $\mathcal{B}_1$'s challenger selects $b = 1$ and stores $\mathbf{F}[\mathrm{t}]$ then a successful $\mathcal{A}_1$ will detect this and correctly output $i' = \mathrm{t}$, and $\mathcal{B}_1$ also wins its game. If $\mathcal{B}_1$'s challenger does not store $\mathbf{F}[\mathrm{t}]$ then $\mathcal{A}_1$ is playing its game with none of the files stored. This is something that it will not expect and thus we need to invoke our assumption that all adversaries in the PEH games terminate. In this case $\mathcal{A}_1$ can do no better than a $\frac{1}{\mathsf{N}}$ guess.

$\mathcal{B}_1$ playing $\mathbf{Exp}^{\mathsf{PEH_d\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{B}_1}$:

$\mathbf{F} \leftarrow \mathcal{A}_1$
$\mathrm{t} \xleftarrow{\$} \{1, \ldots, \mathsf{N}\}$
Give $\mathbf{F}[\mathrm{t}]$ to chall.
for $j \in \{1, \ldots, \mathsf{N}\} \setminus \mathrm{t}$ **do**
  $\mathsf{thr}_{\mathbf{F}[j]} \leftarrow \mathsf{DS.Alg}(\mathbf{F}[j])$
  $\mathsf{ctr}_{\mathbf{F}[j]} \leftarrow 0$
$i' \leftarrow \mathcal{A}_1^{O.\mathsf{PEH.Store}}$
if $i' = \mathrm{t}$ **then**
  return 1
else
  return 0

O.PEH.Store(j) :
if $j = \mathrm{t}$ **then**
  $\mathrm{sig} \leftarrow O.\mathsf{PEH.Store(j)}$
else
  $\mathsf{ctr}_{\mathbf{F}[j]} \leftarrow \mathsf{ctr}_{\mathbf{F}[j]} + 1$
  if $\mathsf{ctr}_{\mathbf{F}[j]} < \mathsf{thr}_{\mathbf{F}[j]}$ **then**
    $\mathrm{sig} \leftarrow 1$
  else
    $\mathrm{sig} \leftarrow 0$
return sig

Figure 5: Reduction $\mathcal{B}_1$ for proof of Theorem 1.

$$\begin{aligned}
&\mathbf{Adv}^{\mathsf{PEH_d\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{B}_1} \\
&\quad = 2 \cdot \left[\mathbf{Pr}\left[\mathbf{Exp}^{\mathsf{PEH_{d1}\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{B}_1} = 1\right] - \frac{1}{2} \right] \\
&\quad = 2 \cdot \left[\mathbf{Pr}\left[\mathcal{B}_1 \Rightarrow 1 \cap b = 1\right] + \mathbf{Pr}\left[\mathcal{B}_1 \Rightarrow 0 \cap b = 0\right] - \frac{1}{2} \right] \\
&\quad = 2 \cdot \left[\frac{1}{2}|\mathbf{Pr}\left[\mathcal{B}_1 \Rightarrow 1 | b = 1\right] + \frac{1}{2}\mathbf{Pr}\left[\mathcal{B}_1 \Rightarrow 0 | b = 0\right] - \frac{1}{2} \right] \\
&\quad = 2 \cdot \left[\frac{1}{2}\mathbf{Pr}\left[\mathbf{Exp}^{\mathsf{PEH_N\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{A}_1} = 1\right] + \frac{1}{2}(1 - \frac{1}{\mathsf{N}}) - \frac{1}{2} \right] \\
&\quad = 2 \cdot \left[\frac{1}{2}(\frac{\mathsf{N}-1}{\mathsf{N}}\mathbf{Adv}^{\mathsf{PEH_N\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{A}_1} + \frac{1}{\mathsf{N}}) - \frac{1}{2\mathsf{N}} \right] \\
&\quad = \frac{\mathsf{N}-1}{\mathsf{N}}\mathbf{Adv}^{\mathsf{PEH_N\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{A}_1}.
\end{aligned}$$

□

We now prove another relation to show the equivalence of the decisional variants, $\mathsf{PEH_d}$ and $\mathsf{PEH_{dN}}$. Clearly one direction is trivial since $\mathsf{PEH_d}$ is a sub-case of $\mathsf{PEH_{dN}}$ so we focus on the converse direction.

Theorem 2. *Let* DS.Alg *be a deduplication threshold selection strategy. For any adversary $\mathcal{A}_2$ against $\mathsf{PEH_{dN}}$ then there exists an adversary $\mathcal{B}_2$ of comparable computational complexity against*

$\mathsf{PEH_d}$ *such that:*

$$\mathbf{Adv}^{\mathsf{PEH_{dN}\text{-}CSSA}}_{\mathcal{A}_2} = \mathbf{Adv}^{\mathsf{PEH_d\text{-}CSSA}}_{\mathcal{B}_2}.$$

Proof. The reduction proceeds in a similar manner to Theorem 1 and is detailed in Fig. 6. b is the challenge bit in $\mathcal{B}_2$'s game, b' is the bit output by $\mathcal{B}_2$ and b'' is the bit output by $\mathcal{A}_2$. Recall that a successful $\mathcal{A}_2$ can tell if one of its N files was stored or not but it doesn't matter which file was stored. This means that as long as $\mathcal{B}_2$ can successfully simulate the game that $\mathcal{A}_2$ is playing, the reduction is perfect. Note that this reduction does not depend on N since it does not matter how many additional files $\mathcal{B}_2$ needs to simulate.

$\mathcal{B}_2$ playing $\mathbf{Exp}^{\mathsf{PEH_d\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{B}_2}$:

$\mathbf{F} \leftarrow \mathcal{A}_2$
$\mathrm{t} \xleftarrow{\$} \{1, \ldots, \mathrm{N}\}$
Give $\mathbf{F}[\mathrm{t}]$ to chall.
for $j \in \{1, \ldots, \mathrm{N}\} \setminus \mathrm{t}$ **do**
  $\mathsf{thr}_{\mathbf{F}[j]} \leftarrow \mathsf{DS.Alg}(\mathbf{F}[j])$
  $\mathsf{ctr}_{\mathbf{F}[j]} \leftarrow 0$
$b'' \leftarrow \mathcal{A}_2^{O.\mathsf{PEH.Store}}$
if $b'' = 1$ **then**
  return 1
else
  return 0

$O.\mathsf{PEH.Store}(j)$:

if $j = \mathrm{t}$ **then**
  $\mathrm{sig} \leftarrow O.\mathsf{PEH.Store}(\mathrm{j})$
else
  $\mathsf{ctr}_{\mathbf{F}[j]} \leftarrow \mathsf{ctr}_{\mathbf{F}[j]} + 1$
  if $\mathsf{ctr}_{\mathbf{F}[j]} < \mathsf{thr}_{\mathbf{F}[j]}$ **then**
    $\mathrm{sig} \leftarrow 1$
  else
    $\mathrm{sig} \leftarrow 0$
return sig

Figure 6: Reduction $\mathcal{B}_2$ for proof of Theorem 2.

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{PEH_d\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{B}_2} &= 2 \cdot \left[\mathbf{Pr}\left[\mathbf{Exp}^{\mathsf{PEH_d\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{B}_2} = 1 \right] - \frac{1}{2} \right] \\
&= 2 \cdot \left[\mathbf{Pr}\left[\mathbf{Exp}^{\mathsf{PEH_{dN}\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{A}_2} = 1 \right] - \frac{1}{2} \right] \\
&= \mathbf{Adv}^{\mathsf{PEH_{dN}\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{A}_2}.
\end{aligned}$$

□

We move on to the relationship between $\mathsf{PEH_N}$ and $\mathsf{PEH_{ind}}$. Clearly one direction is a sub-case; Thm. 3 shows that the converse direction yields the expected security tightness loss.

Theorem 3. *Let* DS.Alg *be a deduplication threshold selection strategy. For any adversary $\mathcal{A}_3$ against $\mathsf{PEH_N}$ then there exists an adversary $\mathcal{B}_3$ of comparable computational complexity against $\mathsf{PEH_{ind}}$ such that:*

$$\mathbf{Adv}^{\mathsf{PEH_N\text{-}CSSA}}_{\mathcal{A}_3} = \frac{\mathrm{N}}{2(\mathrm{N}-1)} \mathbf{Adv}^{\mathsf{PEH_{ind}\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{B}_3}.$$

Proof. The reduction is detailed in Fig. 7. $\mathcal{B}_3$ chooses two files from $\mathcal{A}_3$'s vector of N files and submits them to its own challenger, ensuring that one is stored. It then simulates all the other files as being not stored. If $\mathcal{A}_3$'s index guess does not correspond to one of the two files that $\mathcal{B}_3$ picked then $\mathcal{B}_3$ must simply guess (bit b'' in reduction). b is the challenge bit in $\mathcal{B}_3$'s game, b' is the bit output

$\mathcal{B}_3$ playing $\mathbf{Exp}^{\mathsf{PEH_{ind}\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{B}_3}$:

$\mathbf{F} \leftarrow \mathcal{A}_3$
$\mathrm{t}_0, \mathrm{t}_1 \xleftarrow{\$} \{1, \ldots, \mathrm{N}\}$
$T \leftarrow \{\mathrm{t}_0, \mathrm{t}_1\}$
Give $\mathbf{F}[\mathrm{t}_0], \mathbf{F}[\mathrm{t}_1]$ to chall.
for $j \in \{1, \ldots, \mathrm{N}\} \setminus T$ **do**
  $\mathsf{thr}_{\mathbf{F}[j]} \leftarrow \mathsf{DS.Alg}(\mathbf{F}[j])$
  $\mathsf{ctr}_{\mathbf{F}[j]} \leftarrow 0$
$i' \leftarrow \mathcal{A}_3^{O.\mathsf{PEH.Store}}$
if $i' = \mathrm{t}_0$ **then**
  return 0
if $i' = \mathrm{t}_1$ **then**
  return 1
else
  $b'' \xleftarrow{\$} \{0, 1\}$
  return b''

$O.\mathsf{PEH.Store}(j)$:

if $j \in \{\mathrm{t}_0, \mathrm{t}_1\}$ **then**
  $\mathrm{sig} \leftarrow O.\mathsf{PEH.Store}(\mathrm{j})$
else
  $\mathsf{ctr}_{\mathbf{F}[j]} \leftarrow \mathsf{ctr}_{\mathbf{F}[j]} + 1$
  if $\mathsf{ctr}_{\mathbf{F}[j]} < \mathsf{thr}_{\mathbf{F}[j]}$ **then**
    $\mathrm{sig} \leftarrow 1$
  else
    $\mathrm{sig} \leftarrow 0$
return sig

Figure 7: Reduction $\mathcal{B}_3$ for proof of Theorem 3.

by $\mathcal{B}_3$. This means that $\mathcal{B}_3$ can win in two ways: either $\mathcal{A}_3$ assisted, or b' was guessed correctly after $\mathcal{A}_3$ gave an incorrect index.

$$\begin{aligned}
&\mathbf{Pr}\left[\mathbf{Exp}^{\mathsf{PEH_{ind}\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{B}_3} = 1 \right] \\
&\quad = \mathbf{Pr}\left[\mathbf{Exp}^{\mathsf{PEH_N\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{A}_3} = 1 \right] + \mathbf{Pr}\left[\mathcal{A}_3 \text{ loses but } b'' = b \right] \\
&\quad = \mathbf{Pr}\left[\mathbf{Exp}^{\mathsf{PEH_N\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{A}_3} = 1 \right] \\
&\qquad + \mathbf{Pr}\left[\mathbf{Exp}^{\mathsf{PEH_N\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{A}_3} = 0 \cap b'' = b \right] \\
&\quad = \mathbf{Pr}\left[\mathbf{Exp}^{\mathsf{PEH_N\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{A}_3} = 1 \right] + \frac{1}{2}\left(\frac{\mathrm{N}-2}{\mathrm{N}} \right) \qquad (1) \\
&\quad = \mathbf{Pr}\left[\mathbf{Exp}^{\mathsf{PEH_N\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{A}_3} = 1 \right] + \frac{\mathrm{N}-2}{2\mathrm{N}}.
\end{aligned}$$

So we plug this into the advantage statement equations:

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{PEH_{ind}\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{B}_3} &= 2 \cdot \left[\mathbf{Pr}\left[\mathbf{Exp}^{\mathsf{PEH_{ind}\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{B}_3} = 1 \right] - \frac{1}{2} \right] \\
&= 2 \cdot \left[\mathbf{Pr}\left[\mathbf{Exp}^{\mathsf{PEH_N\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{A}_3} = 1 \right] + \frac{\mathrm{N}-2}{2\mathrm{N}} - \frac{1}{2} \right] \\
&= 2 \cdot \left[\frac{\mathrm{N}-1}{\mathrm{N}} \mathbf{Adv}^{\mathsf{PEH_N\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{A}_3} + \frac{1}{\mathrm{N}} + \frac{\mathrm{N}-2}{2\mathrm{N}} - \frac{1}{2} \right] \\
&= \frac{2(\mathrm{N}-1)}{\mathrm{N}} \mathbf{Adv}^{\mathsf{PEH_N\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{A}_3}.
\end{aligned}$$

□

Theorem 4. *Let* DS.Alg *be a deduplication threshold selection strategy. For any adversary $\mathcal{A}_4$ against $\mathsf{PEH_d}$ then there exists an adversary $\mathcal{B}_4$ of comparable computational complexity against $\mathsf{PEH_N}$ such that:*

$$\mathbf{Adv}^{\mathsf{PEH_d\text{-}CSSA}}_{\mathcal{A}_4} = (\mathrm{N}-1) \cdot \mathbf{Adv}^{\mathsf{PEH_N\text{-}CSSA}}_{\mathcal{B}_4}.$$

$\mathcal{B}_4$ playing $\mathbf{Exp}^{\mathsf{PEH_N\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{B}_4}$:

$\mathsf{F} \leftarrow \mathcal{A}_4$
$\mathsf{F}_0 \leftarrow \mathsf{F}$
$\mathsf{F}_1, \ldots, \mathsf{F}_{\mathsf{N}-1} \xleftarrow{\$} \mathbb{F}$
$\mathbf{F} \leftarrow \mathsf{F}_0, \ldots, \mathsf{F}_{\mathsf{N}-1}$
Give $\mathbf{F}$ to chall.
$b' \leftarrow \mathcal{A}_4^{O.\mathsf{PEH.Store}}$
if $b' = 1$ **then**
  return 0
else
  $i'' \xleftarrow{\$} [1, \ldots, \mathsf{N}-1]$
  return i''

$O.\mathsf{PEH.Store}(j)$:

$\mathsf{sig} \leftarrow O.\mathsf{PEH.Store}(\mathrm{j})$
return sig

Figure 8: Reduction $\mathcal{B}_4$ for proof of Theorem 4.

Proof. The reduction is detailed in Fig, 8. Firstly $\mathcal{A}_4$ outputs a file F. $\mathcal{B}_4$ then randomly picks $\mathsf{N}-1$ files and sends these N files to its challenger, who will store one of them. One can think of the real file F and the first 'fake' file F_1 as being the simulation of the $\mathsf{PEH_d}$ game, since the file F that $\mathcal{A}_4$ is concerned with is stored with probability $\frac{1}{2}$. The other $\mathsf{N}-2$ files reduce the efficacy of this reduction. In this reduction $\mathcal{B}_4$ does not need to simulate $O.\mathsf{PEH.Store}$ queries since the only valid query $\mathcal{A}_4$ can make is on F.

$$\begin{aligned}
&\mathbf{Pr}\left[\mathbf{Exp}^{\mathsf{PEH_N\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{B}_4} = 1\right] \\
&\quad = \frac{1}{\mathsf{N}}\mathbf{Pr}\left[\mathbf{Exp}^{\mathsf{PEH_d\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{A}_4} = 1\right] + \mathbf{Pr}\left[\mathcal{A}_4 \text{ loses but } i'' = i\right] \\
&\quad = \frac{2}{\mathsf{N}}\mathbf{Pr}\left[\mathbf{Exp}^{\mathsf{PEH_d\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{A}_4} = 1\right].
\end{aligned}$$

Plugging this into the advantage statement equations:

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{PEH_N\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{B}_4} &= \frac{\mathsf{N}}{\mathsf{N}-1} \cdot \left[\mathbf{Pr}\left[\mathbf{Exp}^{\mathsf{PEH_N\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{B}_4} = 1\right] - \frac{1}{\mathsf{N}}\right] \\
&= \frac{\mathsf{N}}{\mathsf{N}-1} \cdot \left[\frac{2}{\mathsf{N}}\mathbf{Pr}\left[\mathbf{Exp}^{\mathsf{PEH_d\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{A}_4} = 1\right] - \frac{1}{\mathsf{N}}\right] \\
&= \frac{\mathsf{N}}{\mathsf{N}-1} \cdot \left[\frac{2}{\mathsf{N}}\left(\frac{1}{2}[\mathbf{Adv}^{\mathsf{PEH_d\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{A}_4} + 1]\right) - \frac{1}{\mathsf{N}}\right] \\
&= \frac{1}{\mathsf{N}-1}\mathbf{Adv}^{\mathsf{PEH_d\text{-}CSSA}}_{\mathsf{DS.Alg},\ \mathcal{A}_4}.
\end{aligned}$$

□

4 PLAINTEXT HIDING IN FULL CLOUD STORAGE PROTOCOLS

So far we have discussed and extended the work of HPS and ABDGT on choosing the deduplication threshold. We now consider the wider implications of this side channel as part of fully-fledged cloud storage protocols. In particular we consider this procedure to consist of two distinct parts: an upl.Client procedure that reflects what the client does locally and upl.Server that represents the server's activity in updating its storage state. As mentioned previously, in deployed systems that for the sake of saving bandwidth typically prefer client-side, rather than server-side, deduplication, uploading will often be a three-stage protocol: first the client sends a hash; the server says upload or not; client sends either the file or nothing. In this straightforward scenario the upl.Server procedure will act exactly like the PEH.Store oracle used in Section 3.1, and thus the results detailed there precisely encapsulate plaintext-existence hiding as an attack vector. For more complex systems, the upload procedure may not be the only side channel when it comes to inferring storage status of a file.

In secure deduplication protocols, some key derivation procedure fkeyGen is required to produce keys that, when used, produce deduplicatable ciphertexts. This may involve deriving the file encryption key from the file alone [2, 15] or some other mechanism (such as a third party or additional protocol) to transport the encryption key between valid users [3, 6–8, 10, 14]. If this fkeyGen procedure is an interactive algorithm then this may be another vector of attack for an adversarial client. An example of this being an issue is the protocol of Liu et al. (CCS '15) [8], noted in a revision to the ePrint version [9] of their paper and also in subsequent work by some of the same authors [10]. In their protocol, the file key is determined at the point of the first upload of a file. This is by design but it means that the deduplication threshold is necessarily linked to the number of users that are currently online, rather than the total number of users to store that particular file. This means that a malicious user Eve can choose a time at which she thinks Alice will be online, run the fkeyGen protocol once to acquire some file key fk, abort before uploading the ciphertext, take that client offline and run fkeyGen again. If the two keys are the same then Alice was online and her key material was 'transported' to Eve; if not then the file was not previously stored (or Alice is offline). This attack is particularly interesting because the threshold selection algorithm is – according to ABDGT's analysis – perfect, yet it is another component of the protocol that leaks file storage status.

It is therefore natural to consider a security experiment combining these two attack vectors. The exact security of any concrete protocol in this definitional framework is beyond the scope of this paper, however regarding plaintext-existence hiding in terms of complete cloud storage protocols is an interesting avenue worth pursuing. We thus consider creation of generic results in this space as an open research topic.

Consider an adversary that is attempting to learn whether some file has previously been stored or not (i.e. $\mathsf{PEH_d}$), but that has control over a more generic store procedure than the one considered in the previous section. In particular such an adversary takes into account the potential leakage inferred by the fkeyGen procedure. In this game, the adversary ($\mathcal{A}$) can create valid users and has access to oracles that simulate the interactions those users can make with the protocol(s) in question. In particular, $\mathcal{A}$ can store and delete arbitrary (valid) files, and interact with an O.fkeyGen oracle. To accommodate the secure deduplication protocols mentioned previously that involve inputs from *other users* of the system, $\mathcal{A}$ should of course be able to simulate this by providing such inputs to this oracle. A challenge oracle will take in some file and some user, and either store the file or not. It is then $\mathcal{A}$'s task to work out whether or not the store operation occurred, using its own oracles. In this game, the server's method of processing delete queries raises a number of interesting questions. If there is only one copy

of the file stored, does the server actually remove the file from its backend storage? If so, does it also delete the counter and tag/hash associated with that file? As ABDGT pointed out, this behavior may in fact lead to subtle attacks [[1], § 3.2] and is thus not recommended. This style of security experiment is certainly extensible to the $\mathsf{PEH}_{\mathsf{dN}}$ scenario described earlier.

While it is straightforward to define such a security experiment, it is not immediately obvious if generic results regarding security are possible. For some protocol, if fkeyGen does not take inputs from an external source (i.e. is an algorithm computable using only the file itself) then this attack vector is quashed, and security reduces to that of the threshold selection algorithm. Constructing a truly efficient secure deduplication protocol that is not susceptible to attacks by dishonest clients and an adversarial server remains an open question, and further analysis of this side channel will lead to feasibility results in this space.

5 CONCLUDING REMARKS

In this paper, we investigated a variety of attacks on deduplicating cloud storage systems, in which a (malicious) client attempts to learn the storage status of some plaintext data. We showed that the previously-studied attack scenario – the adversary wishes to learn whether or not a file is stored – is equivalent to an adversary wishing to learn which file from a list is stored. Furthermore, we considered the consequences of such attacks on the upload procedure for deployed cloud storage protocols, and discussed necessary defense mechanisms for thwarting such adversaries. This line of research will allow providers and users to clearly identify which attack vectors exist in deployed systems, and the steps required to mitigate the security risks.

ACKNOWLEDGMENTS

This research was funded by the Research Council of Norway under Project No. 248166.

REFERENCES

[1] Frederik Armknecht, Colin Boyd, Gareth T. Davies, Kristian Gjøsteen, and Mohsen Toorani. 2017. Side Channels in Deduplication: Trade-offs between Leakage and Efficiency. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*. ACM, 266–274. https://doi.org/10.1145/3052973.3053019

[2] Mihir Bellare, Sriram Keelveedhi, and Thomas Ristenpart. 2013. 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, May 26-30, 2013. Proceedings (Lecture Notes in Computer Science)*, Vol. 7881. Springer, 296–312. https://doi.org/10.1007/978-3-642-38348-9_18

[3] Yitao Duan. 2014. Distributed Key Generation for Encrypted Deduplication: Achieving the Strongest Privacy. In *Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security, CCSW '14, Scottsdale, Arizona, USA, November 7, 2014*. ACM, 57–68. https://doi.org/10.1145/2664168.2664169

[4] Danny Harnik, Benny Pinkas, and Alexandra Shulman-Peleg. 2010. Side Channels in Cloud Services: Deduplication in Cloud Storage. *IEEE Security & Privacy* 8, 6 (2010), 40–47. https://doi.org/10.1109/MSP.2010.187

[5] Hermine Hovhannisyan, Kejie Lu, Rongwei Yang, Wen Qi, Jianping Wang, and Mi Wen. 2015. A Novel Deduplication-Based Covert Channel in Cloud Storage Service. In *2015 IEEE Global Communications Conference, GLOBECOM 2015, San Diego, CA, USA, December 6-10, 2015*. IEEE, 1–6. https://doi.org/10.1109/GLOCOM.2014.7417228

[6] Sriram Keelveedhi, Mihir Bellare, and Thomas Ristenpart. 2013. DupLESS: Server-Aided Encryption for Deduplicated Storage. In *Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA, August 14-16, 2013*. USENIX Association, 179–194. https://www.usenix.org/conference/usenixsecurity13/technical-sessions/presentation/bellare

[7] Mingqiang Li, Chuan Qin, Jingwei Li, and Patrick P. C. Lee. 2016. CDStore: Toward Reliable, Secure, and Cost-Efficient Cloud Storage via Convergent Dispersal. *IEEE Internet Computing* 20, 3 (2016), 45–53. https://doi.org/10.1109/MIC.2016.45

[8] Jian Liu, N. Asokan, and Benny Pinkas. 2015. 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*. ACM, 874–885. https://doi.org/10.1145/2810103.2813623

[9] Jian Liu, N. Asokan, and Benny Pinkas. 2015. Secure Deduplication of Encrypted Data without Additional Independent Servers. *IACR Cryptology ePrint Archive* 2015 (2015), 455. http://eprint.iacr.org/2015/455

[10] Jian Liu, Li Duan, Yong Li, and N. Asokan. 2018. Secure Deduplication of Encrypted Data: Refined Model and New Constructions. In *Topics in Cryptology - CT-RSA 2018 - The Cryptographers' Track at the RSA Conference 2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings*. 374–393. https://doi.org/10.1007/978-3-319-76953-0_20

[11] Martin Mulazzani, Sebastian Schrittwieser, Manuel Leithner, Markus Huber, and Edgar R. Weippl. 2011. Dark Clouds on the Horizon: Using Cloud Storage as Attack Vector and Online Slack Space. In *20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings*. USENIX Association. http://static.usenix.org/events/sec11/tech/full_papers/Mulazzani6-24-11.pdf

[12] Tobias Pulls. 2011. (More) Side Channels in Cloud Storage - Linking Data to Users. In *Privacy and Identity Management for Life - 7th IFIP WG 9.2, 9.6/11.7, 11.4, 11.6/PrimeLife International Summer School, Trento, Italy, September 5-9, 2011, Revised Selected Papers (IFIP Advances in Information and Communication Technology)*, Vol. 375. Springer, 102–115. https://doi.org/10.1007/978-3-642-31668-5_8

[13] Young-joo Shin, Dongyoung Koo, and Junbeom Hur. 2017. A Survey of Secure Data Deduplication Schemes for Cloud Storage Systems. *ACM Comput. Surv.* 49, 4 (2017), 74:1–74:38. https://doi.org/10.1145/3017428

[14] Jan Stanek, Alessandro Sorniotti, Elli Androulaki, and Lukas Kencl. 2014. A Secure Data Deduplication Scheme for Cloud Storage. In *Financial Cryptography and Data Security - 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers (Lecture Notes in Computer Science)*, Vol. 8437. Springer, 99–118. https://doi.org/10.1007/978-3-662-45472-5_8

[15] Mark W. Storer, Kevin M. Greenan, Darrell D. E. Long, and Ethan L. Miller. 2008. Secure data deduplication. In *Proceedings of the 2008 ACM Workshop On Storage Security And Survivability, StorageSS 2008, Alexandria, VA, USA, October 31, 2008*. ACM, 1–10. https://doi.org/10.1145/1456469.1456471