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).