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

 ISBN 978-1-4503-4944-4/17/04…$15.00

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-