(More) Side Channels in Cloud Storage

Tobias Pulls

(More) Side Channels in Cloud Storage
Linking Data to Users

Tobias Pulls

Department of Computer Science
Karlstad University
Karlstad, Sweden
tobias.pulls@kau.se

Abstract. Public cloud storage services are gaining in popularity and several commercial actors are offering their services for users, however, not always with the security and privacy of their users as the primary design goal. This paper investigates side channels in public cloud storage services that allow the service provider, and in some cases users of the same service, to learn who has stored a given file and to profile users' usage of the service. These side channels are present in several public cloud storage services that are marketed as secure and privacy-friendly. Our conclusions are that cross-user deduplication should be disabled by default and that public cloud storage services need to be designed to provide unlinkability of users and data, even if the data is encrypted by users before storing it in the cloud.

1 Introduction

The setting for this paper is that of cloud storage services, where a service provider offers a service for users to store their data in the cloud. Commercial cloud storage services such as Dropbox [8], SpiderOak [21], Tarsnap [22], and Wuala [28] provide storage for millions of users and are gaining in popularity. These cloud storage services are all public clouds, that is, they are available to the general public over the Internet [16]. For so called private clouds [16], in the case of all infrastructure being completely operated by the entity using the infrastructure, the issues raised in this paper are in part or completely non-issues. The focus in this paper is on public clouds, and in particular cloud storage services offered by one party (the service provider) to other parties (users).

For cloud storage to be widely adopted, there has to be assurances towards users that their data remain private, and that uploading their data to the cloud has limited effect on what an adversary can learn about them [11, 12]. In other words, the security and privacy of the users of cloud storage services need to be preserved; otherwise some users will not make the move to the cloud.

Our contributions are as follows:

- We formalise an attack that, when cross-user deduplication (explained in Section 2) takes place, has serious privacy implications for users of cloud storage services. The implication of the attack is amplified when storage is distributed among the users of the service.
- We investigate the impact of the attack on a number of different storage services and related systems of particular interest.
- We consider the implications of linkability of users and their data alone (such as can be the case when cross-user deduplication is not performed and encryption is done prior to uploading a file to a storage service) and what threats there still are to the users' privacy, for example for users of CS2 [13], SpiderOak [21], or Tarsnap [22].

Ultimately, we hope that our overall contribution provides a compelling case for (i) that cross-user deduplication should be disabled by default and (ii) that there needs to be, by design, unlinkability of users and their data in cloud storage services.

The following three sections describe the deduplication technique, related work and our adversary model. Section 5 describes an attack, first in general terms, then its implications for distributed storage services such as Wuala [28]. We also investigate the impact of the attack on a number of different systems. The section ends with a summary of the attack. Next, in Section 6, we look at the effects of linkability of users and data on the privacy of the users of cloud storage services. Finally, we conclude the paper in Section 7.

2 Deduplication

We follow the terminology used by Harnik et al. [11]. Deduplication is a technique used by, among others, storage providers to save storage and bandwidth. Instead of storing multiple copies of a file[1] only the original file is kept and links are created to the original when further uploads of the same file are made. Cross-user deduplication is when deduplication takes place across users, for all users of a service or within some set of users. Deduplication can be either target- or source-based. In the target-based approach the service provider performs deduplication, resulting in saved storage space but no savings in bandwidth usage. For source-based duplication, the users perform the deduplication (aided by client software) and both storage space and bandwidth usage are saved, since the user checks with the storage provider whether a file has already been stored before uploading. This check can be, and commonly is, done by calculating the hash of the file and sending it to the storage service[2].

Figure 1 shows a conceptual overview of source-based cross-user deduplication taking place for Alice and Bob. They have both uploaded the same file (seen

[1] Deduplication can take place on the block- or file-level. In this paper we talk about files, but everything discussed can equally well be applied to blocks.

[2] With the assumption, or hope rather, that there will be no hash collision.

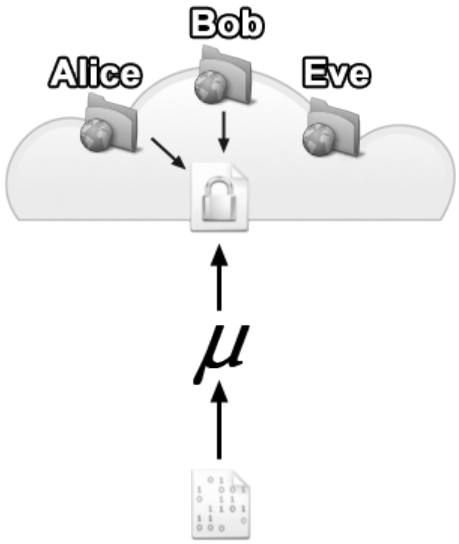

Fig. 1. Source-based cross-user deduplication taking place for Alice and Bob, where the mapping μ of the file to the storage service involves encryption. Only one encrypted copy of the uploaded file is kept by the service provider, and future uploads of the same file by users will only result in references to the previously uploaded file.

at the bottom of the figure), but only one actual copy resides in the storage service. What they both have can be viewed as a folder at the storage service with links to the files they have uploaded. The very first upload of the file to the storage service would naturally involve storing a full copy of the file at the storage provider. However, for later uploads of the same file, by for example Eve, the only change in the service would be that her folder would get a new link to the previously uploaded file. No identical copy of the uploaded file would actually be stored.

When a file is to be stored in the storage service there is some sort of *mapping* taking place, represented in the figure as μ. This mapping can be as simple as encoding the file to the appropriate format, but it might also involve some form of encryption.

Client-side encryption and source-based cross-user deduplication at the same time is made possible ultimately by carefully selecting how, within the set of users for which deduplication is to take place, the encryption key is derived for each file. The function that determines the encryption key for a file has to be based on information that all of the users (for which deduplication is to take place for) has access to. A simple and common approach is to have the encryption key for a file be determined by the hash of the file. In this way, every user that wishes to upload a file will generate the same encryption key and thus produce

the same ciphertext[3] that can easily be deduplicated as described earlier. To be able to decrypt any encrypted file in such a system you would need access to the plaintext in the first place to generate the corresponding key. In Figure 1 the mapping of a file to its representation in the storage service involves some form of encryption. If Eve does not have access to all the information needed to generate the same encryption key as Alice and Bob used, then her encryption of the same file will result in a different ciphertext and appear to the storage service as a different file, preventing cross-user deduplication from taking place.

3 Related Work

Kamara and Lauter [12] provide an overview of what currently can be accomplished by using cryptography when constructing a secure cloud storage service. They also present a practical and provably secure cryptographic cloud storage system named CS2 [13]. In general, cryptographically secure cloud storage services do not use deduplication (we will later explore some exceptions). Deduplication is a technique, as described earlier, to save resources in the form of storage and bandwidth. As with most things in life, there is no such thing as a free lunch. This trade-off was first discussed in academia, to the best of our knowledge, in [7]. Further discussions on the subject can be found online, for example on *The Cryptography Mailing List*[4], initiated by one of the authors of Tahoe-LAFS [26] (Tahoe-LAFS is discussed later in this paper). A relatively recent academic paper on the potential security and privacy issues with deduplication was published by Harnik et al. [11]. Discussion on this trade-off has also led to some storage providers taking an active stance against cross-user deduplication. A prime example is SpiderOak [21] that in a blogpost[5] described in essence the work of Harnik et al. [11] and what a large part of this paper discusses; namely that cross-user deduplication coupled with legal requests to the service provider puts the privacy of the users of the service at risk. Our contribution is, however, more comprehensive: we formalise the attack in question, break it down into its two main components, investigate how it is applicable to more services and systems than SpiderOak, and furthermore describe privacy issues in services, such as SpiderOak, only due to linkability of users and data in Section 6.

When source-based cross-user deduplication takes place, Harnik et al. identified that the storage service in essence is an oracle that answers the query "did any user previously upload a copy of this file?" Utilising this, they presented three side channels in cloud storage, summarised briefly as follows:

[3] Assuming that a deterministic encryption algorithm is used. There are of course other options to this part of the problem, but we opt to exclude all the details and focus on capturing the essence for the sake of clarity.

[4] `http://www.mail-archive.com/cryptography@metzdowd.com/msg08949.html`, accessed 2011-11-09.

[5] `https://spideroak.com/blog/20100827150530`, accessed 2011-11-10.

1. *Identifying Files.* By simply uploading a file and observing if deduplication takes place it is possible to tell if the file has been previously uploaded or not.
2. *Learning the Contents of Files.* By applying the first attack on multiple versions of the same file, an attacker can essentially perform a brute-force attack over all possible values of a file's content.
3. *A Covert Channel.* By uploading specific files deduplication can act as a covert channel for communication for two or more users.

In the same paper, the authors present a solution that mitigates the attacks for a fraction of the files on a storage service by introducing *dummy uploads* for files up to a randomly selected threshold of copies. The size of the fraction of files mitigated depends on the size of the space the threshold is selected from. For uploads up to the threshold target-based deduplication is used. This mitigation comes at the cost of bandwidth but with no negative impact on the storage gain from deduplication for the storage provider.

4 Adversary Model

Soghoian [20] describes the legal precedent in the US for how service providers offering cloud computing services can be forced to turn against their users. This includes obligations to covertly backdoor their software to aid government agencies, as Hushmail was compelled to do in 2007, or enable logging of data available to the service provider, as TorrentSpy was forced to in 2007. Another prime example is the USA PATRIOT Act [23]. It provides law enforcement in the US with the legal right to, among other things, search through electronic communication information and other transaction information at service providers with little restrictions. In Europe, the Data Retention Directive 2006/24/EC [9] requires that communication service providers retain traffic and location data about their users. These examples do not consider what, for example, totalitarian regimes, military organisations or intelligence agencies are capable of. Therefore adversaries to a user include the service provider, and also the other users of the same service. The service provider can itself turn malicious, be honest but curious, or forced through legal means to turn against its users [20]. A service provider could also be required to retain some data by law that could negatively affect the security or privacy of the user [1]. Other users may exploit the design of the storage service offered by the service provider, as shown in [11]. These are all tangible threats to a user of a service, even if the service provider has the best of intentions.

In essence, the goal is to limit the amount of information adversaries can deduce about a user. Any channel that increases the information an adversary has about the user needs to be addressed. Solutions need to be technological in nature and not based upon regulations or policies, since our definition of an adversary includes entities that are obliged by law to invade or restrict the privacy of a user.

5 Linking Files and Users

One, if not the primary, reason for performing deduplication is that it saves significant storage space for the storage provider. This again is presumably greatly enhanced if the deduplication takes place across users (cross-user deduplication). As a consequence of deduplication, the amount of copies stored at the provider of a specific file is reduced, optimally (from a storage point of view) only unique files need to be stored. To accomplish deduplication in storage services, two operations have to be deterministic; the mapping of a file to its representation in the storage service and the function that determines the storage location of a file. If the mapping is non-deterministic, then two uploads of the same file will potentially result in different representations of the file in the storage service. Similarly, if the same representation of a file is not stored at the same place then there will be several copies of the same file stored in the storage service.

In this section, we formalise an attack that, when cross-user deduplication takes place, has serious impact on the privacy of the users of the storage service, by allowing an adversary to link files to users. This attack is not mitigated by the solution presented in [11]. Which adversaries are capable of performing this attack depends on the design of the storage service. After describing the attack we will investigate its applicability in and relation to Wuala, BitTorrent, Freenet and Tahoe-LAFS.

5.1 A Formalised Attack

We denote the mapping of a file f to its representation f' in the storage service as $\mu : f \to (\kappa_{f'}, f')$, where $\kappa_{f'}$ is the lookup key for f' in the storage service. This is the same mapping as was described earlier in Section 2. The storage location for f' in the storage service is denoted as $\lambda_{f'}$ and the function that determines the location in the storage service as $\sigma : \kappa_{f'} \to \lambda_{f'}$. Figure 2 visualises the notation. First the file is mapped by μ into its internal representation (which may involve encryption, as discussed in Section 2). The lookup key for the mapped file is given to the function σ to determine the storage location for the file in the storage service.

Using the notation we just defined, there are two steps involved in being able to link files and users:

1. Acquire a copy of the file f and its mapping $\mu : f \to (\kappa_{f'}, f')$.
2. Obtain the ability to observe requests by users for f'. This can be done, for instance, by monitoring requests to
   (a) the *function* $\sigma : \kappa_{f'} \to \lambda_{f'}$ for the storage location of f', or
   (b) the actual storage *location* $\lambda_{f'}$ of f'.

The mapping μ has to be the same among the users for which deduplication is taking place, otherwise the same file will map to different files within the storage provider and deduplication will be impossible. In other words, for a file f if two different mappings μ_1 and μ_2 are used, then $\mu_1(f) = (\kappa_{f'_1}, f'_1)$ and

Fig. 2. A visualisation of the notation used to describe the linkability attack. The figure shows how a file is mapped into its internal representation in the storage service by the mapping μ, and then its storage location is determined by the function σ.

$\mu_2(f) = (\kappa_{f'_2}, f'_2)$, where $\kappa_{f'_1} \neq \kappa_{f'_2}$ and $f'_1 \neq f'_2$. Since the function σ determines the storage location, and different files should be stored at different locations, then $\sigma(\kappa_{f'_1}) \neq \sigma(\kappa_{f'_2})$ and deduplication does not occur. So, as a consequence of deduplication, step 1 is always possible within the set of users where deduplication is done. Note that this is independent of whether deduplication is target- or source-based. If cross-user deduplication is enabled by default, this means that all users by default use the same mapping and thus can all perform the first step of the attack. Presumably this is also the case for the service provider, who can simply take the role of a user of its service to learn the mapping.

Performing the second step of the attack depends on the setting and design of the storage service. In what is probably the most common setting, where the storage service is being offered by a centralised provider, the storage provider determines where each file should be stored (step 2a) and has the capability of monitoring requests to the actual storage location (step 2b), since it is the entity doing the actual storage. While a storage provider might make claims of not saving this information, if it remains technically possible, a provider might be forced to, as discussed in Section 4.

We have broken down the steps in linking files and users for storage services in terms of and in relation to the essential functionality for storage services performed by the two functions μ and σ. Next, we will look at how feasible it is to link files and users in several settings, starting with the storage service Wuala [28] where storage is distributed among users.

5.2 Wuala - Distributed Storage Among Users

Wuala's distributed storage is an abstraction of a DHT (Distributed Hash Table), broken down into client, storage and super nodes. Super nodes perform all routing by being connected to client and storage nodes. According to available sources, users are client nodes and can only become storage nodes by opting-in to share some local storage. Super nodes are run by Wuala. When a file is uploaded to Wuala it is first encrypted (the key is the hash of the file) and then split into n redundant fragments using erasure codes such that any $m < n$ fragments can be used to reconstruct the file. The fragments are then uploaded to storage

nodes and Wuala's own servers. Information on the design of Wuala is limited to what can be found on Wuala's homepage [28], in a published paper on their cryptographic tree structure [10] and in a Google Tech Talk[6]. Wuala publicly states[7] that it performs source-based cross-user deduplication, which we verified using the procedure outlined in [11].

The next step is translating what we now know about Wuala into the notation used in Section 5.1. Since Wuala performs source-based cross-user deduplication the mapping μ is available and shared by all users. The function σ is run by the super nodes which in turn are run by Wuala exclusively to the best of our knowledge. However, users can query σ to learn $\lambda_{f'}$, where $\lambda_{f'}$ represents the storage locations for all the fragments (or a large enough subset) needed to reconstruct f'. In addition, users can by opting-in become the actual storage location for fragments of files in Wuala.

Since Wuala controls σ and also stores enough fragments to recreate every file uploaded to Wuala, it is clear that they, as the storage provider, have the ability to link any given file in the system to its users. The challenge for a user that wants to perform the same linking lies in becoming a storage node for fragments of the specific file it wants to link. Note that Wuala has an incentive to spread the file fragments among as many storage nodes as possible for the sake of increasing availability, to provide faster downloads for its users and reduce operational costs. We have not attempted to perform this attack on Wuala. The investment in storage and bandwidth required for an attacker is probably feasible for organisations that have a vested interest in linking users that access specific files, such as the media industry fighting piracy.

5.3 BitTorrent - Efficient File Sharing and Linkability

BitTorrent, while not a storage service but a peer-to-peer file sharing protocol, is interesting in relation to the attack described earlier. Le-Blond et al. [15] describe how to spy on most BitTorrent users on the Internet from a single computer. They accomplish this by querying popular BitTorrent trackers[8] or the DHT used by BitTorrent clients. In essence, this can be viewed as querying σ to learn $\lambda_{f'}$, where $\lambda_{f'}$ is a list of *all* users with parts, or the entirety, of f'. This highlights the danger of providing full access to σ and a shared μ to users at the same time. Naturally, when trackers are involved they are able to link users of the tracker and the files they are accessing. The main purpose of using a DHT in BitTorrent is to eliminate the need for a tracker. As a consequence users, beyond the ability they already had to query σ, take part in actually running the functionality σ provides towards other users.

In later paper [14], Le-Blond et al. used the ability to spy on BitTorrent users together with an information leakage in Skype to determine where users

[6] `http://www.youtube.com/watch?v=3xKZ4KGkQY8`, accessed 2011-05-17.

[7] `http://wualablog.blogspot.com/2010/10/top-ten-questions-and-their-answers-how.html`, accessed 2011-05-17.

[8] A BitTorrent tracker can be viewed as a key-value store run as a dedicated service, where keys are hashes of files and values are users that are downloading the file.

are, who they are and what they are sharing. The linkability of users and data, later enriched by correlating with another source of information, turned into a very real privacy threat.

5.4 Freenet - Anonymous Distributed and Decentralised Storage

Freenet is a distributed and decentralised data store that provides a high degree of anonymity [4]. All the users that participate in the system act as distributed storage nodes and are connected, depending on configuration, in a peer-to-peer or friend-to-friend network[9]. Files stored in Freenet are identified by a key and mapped (encrypted) based upon the type of key selected within Freenet. Performing step 1 in the attack described in Section 5.1 is straight forward for any static file (content hash keys in Freenet) but requires knowledge of the generated address for dynamic files (signed subspace keys in Freenet). However, when attempting to perform step 2 of the attack the design of Freenet mitigates both step 2a and 2b. Monitoring requests for the storage location of a file ($\sigma : \kappa_{f'} \rightarrow \lambda_{f'}$) is fruitless, since nodes in Freenet only communicate with their peers. A request from a peer for the storage location of a file is the same as if it was a peer of the requesting peer that requested the file. In the same way, monitoring requests to the actual storage location of a file ($\lambda_{f'}$) is ineffective because one can only tell which peer, and not user, that requested the file. Furthermore, files are *replicated* to other users in Freenet when they are requested or inserted into the system, making the number of storage locations for a popular file increasingly large.

5.5 Tahoe-LAFS - Multiple Storage Providers

Tahoe-LAFS is an open-source storage system that uses multiple storage providers to provide what they refer to as "provider-independent security" [26]. By spreading data over multiple storage providers, using erasure encoding of encrypted data like Wuala, Tahoe-LAFS becomes resistant to compromise of a subset of the providers being used. This resilience comes at the cost of introducing redundancy in storage, which degree can be configured by users. To reduce this redundancy further, Tahoe-LAFS can be *configured to enable* cross-user deduplication within a set of users by setting the same "convergence secret". The convergence secret can be viewed as a salt to the hash of the file used to generate the encryption key for a file. So, by default in Tahoe-LAFS cross-user deduplication is disabled and files are encrypted with unique encryption keys, thus provide different mappings (μ) for each user. Furthermore, the convergent secret is not known to any of the storage providers, so even if cross-user deduplication is enabled by a set of users, as long as they do not share their convergent secret with any service provider a provider will be unable to obtain a mapping of a file, thus preventing the provider from linking any files. To make matters worse for a storage provider that wishes to link files and users, since multiple

[9] A friend-to-friend network is a peer-to-peer network where connections are only made between friends and not to all potential peers [25].

storage providers are used obtaining the convergent secret of a user is not enough since the provider in question may not be the provider storing a particular file of interest for the user.

5.6 Summary

When cross-user deduplication takes place the number of storage locations for a file is minimised to save storage. Furthermore, the mapping of a file to its representation in the storage service has to be the same for all users that cross-user deduplication is taking place for. When cross-user deduplication is *enabled by default* presumably the storage provider is aware of this mapping as well. This means that the storage provider, unless specifically designed to prevent it, has the ability to link a given file to all users with that file in the storage service. It can do this by simply observing the requests for the given file's storage location or requests to the actual storage location of the file. Also, when storage is distributed among users they can gain the ability to link a given file to users, unless the service has been designed with preventing this in mind, such as is done in Freenet at the cost of bandwidth and storage for all users.

BitTorrent is an efficient peer-to-peer file sharing protocol, with little to no privacy by design for its users. The ease with which users and files are linkable are at the center of its efficiency. This highlights the danger (for users) of prioritising efficiency over security and privacy. The Tahoe-LAFS system puts the security and privacy if its users first by (i) not trusting storage providers, (ii) having cross-user deduplication disabled by default and (iii) letting the users decide about the trade-off between privacy, security and efficiency on their own.

6 Profiling Users' Usage

Even if cross-user deduplication is not performed, linkability is still a privacy problem. We first describe how a storage provider can profile its users by simply observing their stored data and then discuss how to best mitigate this profiling.

6.1 Observing Storage

Cross-user deduplication is not performed (or even possible) when the mapping of a file to its representation in the storage service is different for different users. For example, this is the case when the user encrypts the file with a key that only they posses before uploading it to the storage service. Then neither the storage provider nor other users can link *a given file* to users using the side channels described earlier. However, unless users take special precautions and the storage service is designed to prevent it, the storage provider can link stored (potentially encrypted) *data* and users. This allows the storage provider to profile their users in terms of *when* data is accessed, *how much* data is stored, and *what action* did they take. Also, unless the user takes special precautions, the storage provider can determine from *where* the user accesses the service.

Figure 3 shows an example of the kind of profiling a service provider could do by simply observing storage utilisation over time, which together with knowledge of who performed these actions and from where is non-negligible information. Encryption only hides the actual information stored. As is shown in Figure 3, if the storage provider can tell that a user uploaded a file with the size 1491834576 bytes prior to the release of the Wikileaks insurance file at the end of July 2010 [27] it may pose a threat to that user.

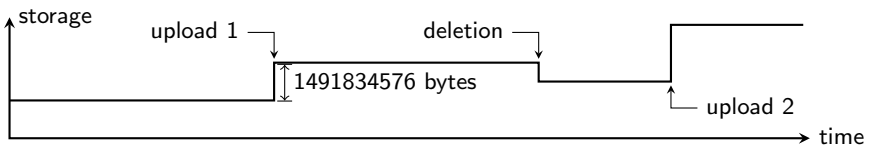

Fig. 3. By observing storage utilisation over time a storage provider can profile a user.

Asking who, what, when, where, how and why is a concept from journalism known as the *Five Ws*; a fundamental concept in gathering information [24]. In fact, observing storage utilisation is similar to performing traffic analysis; both deal with analysing non-content data [5]. Just as traffic patterns may reveal sensitive information so may storage patterns. The need for hiding this kind of activity information in the cloud setting is a known problem for businesses, and may leak, or in and of itself be, confidential business information [3].

6.2 Mitigating Profiling

How much data a user has stored, and what is done with it, can be hidden by the user at the cost of storage and bandwidth, by for example using padding or encrypting a full container of data. When data is accessed can be hidden by introducing dummy traffic and uploads. From where the user is accessing the service can be hidden by using proxies or other anonymisation services. Concealing who the data belongs to on the other hand, which is arguably the most important thing with the user's privacy in mind, requires that the service is designed to support it. Recent work by Slamanig [19] is a good example of a scheme for a service where *who* a resource (such as storage) in the cloud belongs to is cryptographically hidden. Another, however costly, approach to mitigation is to use multiple service providers, as is done in Tahoe-LAFS. Freenet goes a step even further (in terms of cost) and gets rid of the storage provider altogether.

One could argue that most storage services targeting regular users today do not require any proper identification of users to take place (so being pseudonymous is possible), but any significant storage capacity requires the user to pay for the service. In the absence of any widely available way to pay anonymously online (such as using anonymous prepaid cards, like those offered by paysafecard [18], or carefully spending Bitcoins [17]), this enables the service provider to link

the user's storage with their payment details, unless designed to prevent this very linking.

Addressing the linkability on its own may not be enough, since observing the storage utilisation of data stored at the service provider correlated with other information (like identified users performing some action that should lead to changes being made to their storage) may in and of itself be enough to link data and users. However, the effort for the service provider to profile users increases significantly if they first have to determine who the data they are storing belongs to.

7 Conclusion

Without strong assurances to users that their data is private and secure in the cloud, storage services will not be as widely adopted as they could be [11, 12]. Cloud storage services should be designed to provide unlinkability of users and data, even if the data is encrypted by users, and cross-user deduplication should be disabled by default. Cross-user deduplication leads to side channels in cloud storage which can be mitigated by sacrificing bandwidth and storage; the main reasons for performing deduplication in the first place. Furthermore, if the service provider can link the data it stores to users that in and of itself opens up side channels that leak information about the users' usage of the service. Service providers might make claims of not storing, using or sharing information about users, but as long as it remains technically possible they may be compelled to hand over all information they can technically produce to authorities.

Harnik et al. [11] presented a mitigation strategy for their identified side channels. It is possible to conceive of complementary mitigation to the side channels we have described in this paper by using anonymous credentials [2], or the scheme by Slamanig [19], together with the use by users of some anonymisation service, such as Tor [6], for accessing a storage service. This *might* be sufficient for having cross-user deduplication enabled by default. There is a need to further investigate the trade-off imposed by cross-user deduplication, but until then, the only sensible approach is to leave it off by default for the sake of privacy.

Acknowledgments We would like to thank Simone Fischer-Hübner, Stefan Lindskog, Stefan Berthold, Philipp Winter, all the attendees of the IFIP Summer School 2011, and the reviewers for their valuable feedback. This work is funded by a Google research grant on "Usable Privacy and Transparency Tools".

References

1. Berthold, S., Böhme, R., Köpsell, S.: Data retention and anonymity services. In: The Future of Identity in the Information Society. Springer Boston (2009)
2. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: EUROCRYPT. pp. 93–118 (2001)
3. Chen, Y., Paxson, V., Katz, R.H.: What's new about cloud computing security? Tech. Rep. UCB/EECS-2010-5, EECS Department, University of California, Berkeley (Jan 2010), http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-5.html
4. Clarke, I., Miller, S.G., Hong, T.W., Sandberg, O., Wiley, B.: Protecting free expression online with freenet. IEEE Internet Computing 6(1), 40–49 (2002)
5. Danezis, G., Clayton, R.: Introducing traffic analysis. In: Attacks, Defences and Public Policy Issues. CRC Press (2007)
6. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: Proceedings of the 13th USENIX Security Symposium (August 2004)
7. Douceur, J.R., Adya, A., Bolosky, W.J., Simon, D., Theimer, M.: Reclaiming space from duplicate files in a serverless distributed file system. In: ICDCS '02: Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02). p. 617. IEEE Computer Society, Washington, DC, USA (2002)
8. Dropbox: Dropbox - simplify your life, https://www.dropbox.com/, accessed 2011-05-17
9. EUR-Lex - Access to European Union law: 32006l0024 - en (2012), http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32006L0024:EN:NOT, accessed 2012-02-20
10. Grolimund, D., Meisser, L., Schmid, S., Wattenhofer, R.: Cryptree: A folder tree structure for cryptographic file systems. In: Symposium on Reliable Distributed Systems. pp. 189–198 (2006)
11. Harnik, D., Pinkas, B., Shulman-Peleg, A.: Side channels in cloud services: Deduplication in cloud storage. IEEE Security & Privacy 8(6), 40–47 (2010)
12. Kamara, S., Lauter, K.: Cryptographic cloud storage. In: Financial Cryptography and Data Security, Lecture Notes in Computer Science, vol. 6054, pp. 136–149. Springer Berlin / Heidelberg (2010)
13. Kamara, S., Papamanthou, C., Roeder, T.: CS2: A semantic cryptographic cloud storage system. Tech. Rep. MSR-TR-2011-58, Microsoft Technical Report, http://research.microsoft.com/apps/pubs/?id=148632 (May 2011)
14. Le-Blond, S., Chao, Z., Legout, A., Ross, K.W., Dabbous, W.: I know where you are and what you are sharing: Exploiting P2P communications to invade users' privacy. CoRR abs/1109.4039 (2011)
15. Le-Blond, S., Legout, A., Fessant, F.L., Dabbous, W., Kâafar, M.A.: Spying the world from your laptop – identifying and profiling content providers and big downloaders in bittorrent. CoRR abs/1004.0930 (2010)
16. Mell, P., Grance, T.: The NIST definition of cloud computing http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf
17. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2009), http://www.bitcoin.org/sites/default/files/bitcoin.pdf
18. Paysafecard: Discover the prepaid solution for the internet : paysafecard.com, http://www.paysafecard.com, accessed 2011-11-20

19. Slamanig, D.: Efficient schemes for anonymous yet authorized and bounded use of cloud resources. Selected Areas in Cryptograph (2011), to appear
20. Soghoian, C.: Caught in the cloud: Privacy, encryption, and government back doors in the Web 2.0 era. Journal on Telecommunications and High Technology Law 8(2), 359–424 (Spring 2010), http://www.jthtl.org/content/articles/V8I2/JTHTLv8i2_Soghoian.PDF
21. SpiderOak: Zero-knowledge data backup, sync, access, storage and share from any device — spideroak.com, https://spideroak.com/, accessed 2011-11-15
22. Tarsnap: Tarsnap - online backups for the truly paranoid, https://www.tarsnap.com/, accessed 2010-05-17
23. The U.S Government Printing Office: Public Law 107 - 56 - Uniting and Strengthening America by Providing Appropriate Tools Required to Intercept and Obstruct Terrorism (USA PATRIOT ACT) Act of 2001 (2001), http://www.gpo.gov/fdsys/pkg/PLAW-107publ56/content-detail.html, accessed 2012-02-20
24. Wikipedia: Five ws — wikipedia, the free encyclopedia (2011), https://secure.wikimedia.org/wikipedia/en/w/index.php?title=Five_Ws&oldid=442072782, accessed 2011-08-12
25. Wikipedia: Friend-to-friend — wikipedia, the free encyclopedia (2012), http://en.wikipedia.org/w/index.php?title=Friend-to-friend&oldid=474069021, [Online; accessed 18-February-2012]
26. Wilcox-O'Hearn, Z., Warner, B.: Tahoe: the least-authority filesystem. In: Proceedings of the 4th ACM International Workshop on Storage Security and Survivability (StorageSS '08). pp. 21–26. ACM, ACM, New York, NY, USA (2008), http://portal.acm.org/citation.cfm?id=1456474$\#$
27. Wired: Wikileaks posts mysterious 'insurance' file — threat level — wired.com (2011), http://www.wired.com/threatlevel/2010/07/wikileaks-insurance-file/, accessed 2011-08-12
28. Wuala: Wuala - secure online storage - backup. sync. share. access everywhere., https://www.wuala.com/, accessed 2011-05-17