Privacy-aware Data Storage in Cloud Computing

Rémy Pottier and Jean-Marc Menaud
IMT Atlantique, 4 rue Alfred Kastler, 44307 Nantes, France

Keywords: Cloud Computing, Storage, Privacy, Security.

Abstract: The increasing number of cloud storage services like Dropbox or Google Drive allows users to store more and more data on the Internet. However, these services do not give users enough guarantees in protecting the privacy of their data. In order to limit the risk that the storage service scans user documents, for example, for commercial purposes, we propose a storage service that stores data on several cloud providers while prohibiting these providers to read user documents. Indeed, the proposed sky storage service (i.e., a service composed of several cloud services) named SkyStore, protects the user privacy by breaking user documents into blocks and spreading these blocks over many cloud storage providers. The architecture of this service ensures that SkyStore can not read user documents. It connects directly users to cloud providers in order to avoid trusting a third-party. This paper consists of two parts. First, the sky service architecture is described to detail the different protections provided to secure user documents. Second, the consequences of this architecture on the performance are discussed.

1 INTRODUCTION

Nowadays, the cloud computing offers a very wide variety of services like cinema programs, weather forecast or television series streaming. Moreover, these services are readily available for free download from application stores like the Apple App Store or Google Play. These applications, for desktop computers or mobile phones, change the way we interact with online services. In promising to access our documents (pictures, movies, text files) from anywhere, the number of users of cloud storage services like Dropbox grows rapidly. However, storing personal data in the cloud needs trusted storage services that ensure the protection and the privacy of our data. For example, some mail services scan the content of emails for tailored advertising, so it does not appear impossible to consider that some storage services could scan the content of our files in our own best interest, as they say. With storage services like Dropbox, users must have full trust in the cloud provider managing their data because providers can read user documents and retrieve information about users. So, some users subscribe to storage services that use encryption at the user side like SpiderOak. As data is encrypted before outsourcing it to a cloud provider, the user privacy is strengthened because the provider can not read user documents. However, the recent FBI-Apple encryption dispute[1] or successful side-channel attacks (Genkin et al., 2014) have shown us that decrypting data is possible. Moreover, in countries like France, the size of the encryption key is limited to 128 bits with all the consequences that involves for the data privacy.

In order to help users to ensure the privacy of data stored on different cloud providers, we proposed a storage service based on many cloud providers running exclusively on the user side. To ensure the data privacy without using encryption, users break their documents into blocks and these blocks are stored on cloud providers. By ensuring that each cloud provider owns only one part of blocks, cloud providers can not read or deliver user documents. To avoid the use of a trusted third-party that could threaten the data privacy, the storage service is not hosted on the cloud but it runs on user devices and it connects directly to cloud providers. So, user data and the organization of the data on cloud providers, i.e., the metadata, are only managed by users, and so, only users can read their documents. Consequently, the trust is placed in the application on the user device rather than in the storage service on the cloud.

In the remainder of this paper, we describe mechanisms for ensuring the user data privacy in the pro-

[1]http://www.cnbc.com/2016/03/29/apple-vs-fbi-all-you-need-to-know.html

Pottier, R. and Menaud, J-M.
Privacy-aware Data Storage in Cloud Computing.
DOI: 10.5220/0006294204050412
In *Proceedings of the 7th International Conference on Cloud Computing and Services Science (CLOSER 2017)*, pages 377-384
ISBN: 978-989-758-243-1

posed cloud storage service. Section 2 gives an overview of existing works about privacy-aware systems and cloud storage. Section 3 shows the architecture of the storage service and it describes the management of both the data and the metadata associated with user documents. Section 4 analyzes the performance of a prototype of the storage service and it compares this prototype to existing cloud storage services. Section 5 provides a conclusion and future works.

2 RELATED WORKS

In the early 2000s, many works contribute to possible solutions for anonymous storage accessible by the Internet based on peer-to-peer architectures (Dingledine et al., 2000)(Waldman and Mazieres, 2001). These systems permit to store and access documents while protecting the anonymity of users. The cooperation of the high number of nodes in the architecture allows to hide where the data is really stored and the user that stores or reads the document. Tangler and The Free Haven Project break documents into smaller blocks by using redundant distribution schemes, such as erasure code algorithms (Luo et al., 2009) or information dispersal algorithms (Rabin, 1989), in order to efficiently replicate documents. In these peer-to-peer systems, the high number of nodes is critical to ensure a large storage space and the anonymity of both users and documents. Additionally, the system can not guarantee optimum performance because it can not control node capabilities.

For the purpose of providing a reliable storage space on the Internet with a competitive price, storage providers use the possibilities offered by the cloud. To avoid being dependent on one single cloud provider, RACS (Abu-Libdeh et al., 2010) and SCMCS (Singh et al., 2011) propose cloud storage services using several cloud providers to improve the data availability by maintaining redundancy in data distribution and to preserve the data privacy. In these systems, the privacy is achieved by dividing documents into blocks and distributing these blocks among several clouds or providers. In this way, cloud providers only host a part of user documents, and so, it can not read them. To reconstruct the whole document, cloud providers must collude with each other to exchange the parts of the document. In fact, the use of redundant distribution schemes could be used to cut user documents into blocks and shuffle the block content in order to make the document hard to reconstruct (Cincilla et al., 2015). So, even with all data blocks, reading documents is not an easy task. However, the architecture proposed by RACS and SCMCS involves a third-party that knows the block organization. So, users must trust the third-party instead of cloud providers. Although this third-party does not own the user data, it could read user documents before storing them in cloud providers, ask the blocks to cloud providers or give the block organization to cloud providers. In some way, the issue of the data privacy is moved to cloud providers to the third-party on the cloud. In our approach, the third-party does not exist. Users connect directly to several cloud providers with their cloud accounts from the SkyStore application. In this architecture, users do not share critical data like the block organization with anyone but they must trust the application not to release personal data.

In this paper, we explore the possibility of protecting the user privacy from a storage service using several cloud storage providers in the same way as RACS and SCMCS. So, the system breaks user documents into smaller blocks and stores blocks to different providers. The privacy is protected by two main ideas. First, the storage service does not store all the data in the same cloud provider. In this way, the first step in reading documents is to retrieve every block. The second step is to assemble the blocks to reconstruct documents. The storage service shuffles the data before building blocks and cloud providers do not know the shuffle algorithm. So, putting the blocks together is not a straightforward task. To increase the complexity of these two steps, the storage service stores blocks of every user with no organization on cloud providers. So, at the provider level, a large amount of blocks belonging to users is stored with no information. As in peer-to-peer systems, the more blocks there are, the more complicated is to reconstruct documents. Unlike RACS and SCMCS, the information to reconstruct documents, namely the metadata, is not owned by a third-party that does not exist in our architecture. Consequently, users are the only ones who can read the data, i.e., their documents.

3 USER DATA PRIVACY

In our cloud storage service SkyStore, the role of users is far more important than just selecting documents to upload on the cloud. Indeed, users are responsible to break their documents into blocks before sending them to the cloud storage service. This operation, called the encoding process, allows users to choose the algorithm to build blocks and select providers. So, users install the storage service to communicate with cloud providers on their device, e.g., a desktop computer. Then, users choose cloud providers from the list of available ones. At least two providers are required to distribute data blocks but a higher number of providers increases the privacy by decreasing the amount of data stored on each provider. Users also choose the algorithm used to break their documents. By selecting their algorithm, users may choose to use cut-and-shuffle (Cincilla et al., 2015) or erasure code algorithms depending on the characteristics choosen. The cut-and-shuffle algorithm ensures the respect of the user privacy by mixing the data and breaking it into blocks. The erasure code algorithm (Luo et al., 2009) allows the loss of blocks to prevent cloud provider outage but the size of documents is increased. For instance, when users store 100 MB of data on three providers by using the erasure code algorithm, 150 MB of data are stored on cloud providers but only two providers are required to reconstruct the data. So, in case of a single provider failure, user documents can be downloaded.

During the encoding process, metadata is generated at the user side as shown in Figure 1. The metadata is the information to retrieve all blocks of documents and assemble these blocks. The metadata of one document consists of the encoding algorithm, the number of blocks, the block names and the location of blocks, i.e., the cloud providers used to store the blocks of the document. Metadata can not be stored to user devices because of device failures and theft of devices. The loss of metadata means the loss of user documents. So, metadata is stored on cloud providers with the same encoding process than user documents and the required information for retrieving it is computed from functions as detailed in Section 3.2.

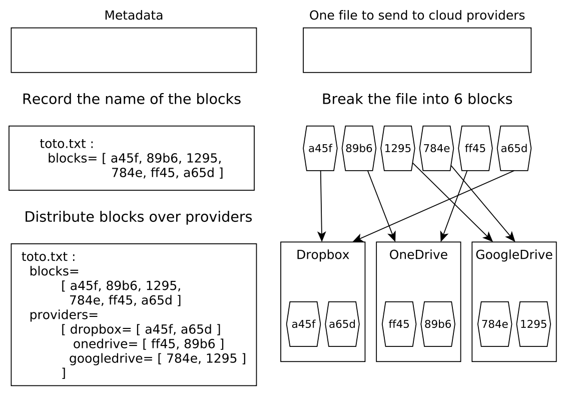

Metadata
One file to send to cloud providers
Record the name of the blocks
Break the file into 6 blocks
toto.txt : blocks= [a45f, 89b6, 1295, 784e, ff45, a65d]
a45f
89b6
1295
784e
ff45
a65d
Distribute blocks over providers
toto.txt : blocks= [a45f, 89b6, 1295, 784e, ff45, a65d] providers= [dropbox= [a45f, a65d] onedrive= [ff45, 89b6] googledrive= [784e, 1295]]
Dropbox
OneDrive
GoogleDrive
a45f
a65d
ff45
89b6
784e
1295

Figure 1: Metadata generation during the encoding process.

3.1 Data Storage

In our previous work, we proposed a cloud storage service, named TrustyDrive (Pottier and Menaud, 2016), based on a traditional architecture. This architecture, also used in above-mentioned works (Abu-Libdeh et al., 2010)(Singh et al., 2011), connects users and cloud providers by means of a third-party. Although the main concern of TrustyDrive is the anonymity, the use of a third-party forces users to trust another component. This component is responsible for managing and storing the metadata, and so, it can exploit or disclose it without user consent that could reduce the privacy of users. In the proposed architecture, users and cloud providers are directly connected and users have access to the storage space of cloud providers.

The first advantage to store blocks of one document on several providers is that each provider has only one part of the data. Moreover, the data is mixed by the encoding algorithm, and so, even one block of a plain text is illegible. As a result, one provider, or someone with an access to the provider, who wants to exploit the data of the user could not do it without retrieving the blocks hosted in other providers. An attacker, or a malicious organization, that wants to retrieve blocks from every provider, has to hack or pressure several providers. Actually, it is easier to operate on a single provider instead of several ones because there are more weakness to find and exploit.

In the improbable case of the attacker succeeds in retrieving every block from providers used by the storage service, he has to reconstruct documents. As the attacker has collected lots of blocks, he could try to limit the number of blocks to decrypt by analyzing only the blocks associated with a specific user or a specific document. However, the cloud providers do not have any information about the block organization, and so, the attacker can not group blocks by user or by document from the hack of the storage service.

In fact, the attacker may have some information about blocks coming from every provider like the creation date and the block size. These information are available from common cloud providers and they could compromise the user privacy.

From the creation date, the attacker could group blocks by date and assume that every block with nearly equal dates belong to the same document. This information threatens the privacy because the attacker can now try to decrypt the document from a small amount of blocks. In common file systems, the creation date of a file could be customized with basic commands (e.g., on Linux, the bash command *touch* could modify date attributes of a file[2]). Unfortunately, such customizations (e.g., modification of the creation date) are not available from cloud provider API like the Dropbox API, and so, the application can not hide this information from attackers.

[2]http://tinyurl.com/nnopzzj

Assuming that the encoding algorithm breaks one document into blocks with a same size, the attacker can also group blocks by their size in order to easily detect blocks belonging to the same document. The privacy is, once again, threatened. To solve this issue, encoding algorithms could make blocks with random sizes but it is easier to design algorithms that create blocks of equal sizes for all documents. As the design of encoding algorithms is already complicated, this option is rejected. Another option is to use blocks with a fixed size, either using padding or making small blocks. In order to have blocks with equal sizes, the system fills blocks with useless data, called padding, until the required size. The main disadvantage of padding is to increase the size of the data to upload for one document. An alternative of padding is to build small blocks with a fixed size to increase the number of blocks with equal sizes on each provider. In that case, the number of blocks generated for one document increases in order to drown blocks associated to one document in a sea of blocks. This solution allows to protect the privacy without uploading more data than necessary but increase dramatically the number of blocks to upload. We choose the last option by counting on simultaneous downloads and uploads of blocks to reduce the loss of performance.

3.2 Metadata Storage

Once users have encoded their documents, the metadata of these documents contains the entire information required to reconstruct them. So, the protection of metadata is a determining factor to preserve the user privacy. Moreover, metadata must be saved in a safe storage because the loss of metadata prevents users to reconstruct their documents. As a computer and a mobile phone can be stolen, the backup of metadata must be stored in a remote storage. In our approach, this remote storage is the sky storage service itself. To minimize the potential damage of a theft, metadata is never written to the local drive of the user device but it remains in the device memory. So, when users close the application, their metadata is destroyed.

The permanent storage of metadata is performed by breaking it into blocks from the encoding algorithm before sending the generated blocks to the cloud storage service. Two pieces of information are required to store the metadata blocks: a list of cloud providers and a list of block names. To store the metadata, all registered providers are used. Consequently, users must obtain access to every provider to retrieve their metadata from other devices. So, users must manually connect to the right providers before retrieving their metadata, and so, their documents. This seems to be restrictive but it is necessary for the security of the application in order to avoid to give cloud provider credentials, mostly passwords, to a third-party. However, credentials are stored on user devices, and so, users have to authenticate just once on cloud providers. An additional password is asked at the storage service startup to protect user documents in case the computer or the phone is stolen. To compute the list of metadata block names, cryptographic functions calculate the block names from the password, the name of the provider and the email of the provider account. For example, the function SHA-1 could be used for calculating names of blocks required to save the metadata of the user from its password as follows: *SHA1(password + providerName + email)*.

In conclusion, the retrieval of metadata is based on (i) the correct list of cloud providers, (ii) the credentials of these cloud accounts, (iii) the block names defined from passwords, emails and provider names of cloud provider accounts.

3.3 Architecture Comparison

In the storage service proposed in this paper, users control their data by choosing the encoding algorithm and by avoiding that cloud providers can read their metadata. In TrustyDrive (Pottier and Menaud, 2016), the anonymity provides a stronger data privacy by hiding the owner of the data but it is achieved by using a third-party but users must trust the cloud storage service for not saving metadata. In the remainder of this section, we discuss the benefits and the disadvantages of connecting users directly to cloud providers or by the means of a third-party.

A direct connection between users and cloud providers means that users create one account on each cloud provider. For an architecture with three cloud providers, the creation and the connection do not take a long time but, for seven or more cloud providers, this operation can quickly become cumbersome and tricky. Moreover, inexperience people may find difficult to know exactly the features of the cloud provider, for example, the transfer rates, the location of the data, etc. This features could be analyzed and proposed by expert third-parties. The last, but not the least, disadvantage is the number of blocks stored on cloud providers. In the architecture without any third-party, cloud providers are used by one single user. So, the number of blocks could not be sufficient to ensure the data privacy and credentials to connect to cloud providers can lead to the user identity.

By using a third-party as a proxy to connect to cloud providers, users only connect to one service. They only have to create one account on the third-party. Moreover, as one third-party is used by many users, the number of blocks stored on cloud providers dramatically increases that improves the data privacy. Indeed, as the third-party of the TrustyDrive architecture can not link users to documents, retrieving all the documents of a specific users is technically unfeasible unless the third-party is corrupted or malicious. However, as users do not own the credentials of cloud providers, they can not have access to the physical storage space to check the content of the blocks, for instance, to check that blocks are illegible. In the architecture described in this paper, users have full access to the storage space, and so, they can check the data stored on every cloud provider.

4 THE COST OF THE PRIVACY

The storage service presented here proposes to store documents on the cloud while ensuring the user privacy. In this section, we analyze the cost associated with uploading and downloading documents from a prototype designed as a privacy-aware storage service.

One part of the extra cost is related to prevent the theft of documents. Unlike other cloud storage services like Dropbox, Google Drive, Microsoft OneDrive, etc., the system does not keep a local copy of documents. So, the theft of the user device (laptop, mobile phone) does not involve the theft of documents but users have to frequently download documents to read or edit them.

Another additional cost comes from both the generation and the transfert of blocks. As data encryption involves a performance overhead, encoding algorithms consume both processor and memory resources. So, the time to build blocks slows down document transfers. Moreover, documents are divided into several blocks that increases the number of upload and download requests to handle documents, and so, times to upload and download documents raise.

Metadata describes the organization of user documents and every modification of this organization, like uploading new documents, renaming or moving documents, modifies the metadata. In common cloud storage services like Dropbox, when a document is uploaded, the only piece of metadata to upload is the document name. The organization of the document data only concerns the cloud storage service. Users does not know how their data is stored (e.g., is the document stored on one single file or on many files ?). As the proposed storage service manages the user metadata on the user side, users must upload it. In addition, the local metadata on the user side is different from the metadata stored on the cloud after every modification of user documents. To maintain the synchronization between the local metadata and the remote one, the application uploads it after every modification. So, metadata is frequently uploaded which has a direct negative impact on the performance of the system.

One parameter that influences the performance of the system is the block size because it defines the number of blocks necessary to encode one document. As mentionned in Section 3.1, this feature of the system is used to generate a sufficient number of blocks in order to ensure the privacy. However, an excessive number of blocks reduces the performance of the system. A balance must be struck between the privacy and the performance of the system.

The next section describes the prototype of a storage service guided by above mentionned principles. In the Section 4.2, the evaluation of the prototype clarifies the impact of the block size on the performance of the system. A comparison between our prototype of the storage service and existing cloud storage service is given in the Section 4.3.

4.1 Prototype Description

Unlike TrustyDrive (Pottier and Menaud, 2016), which chooses an architecture with a third-party as described in the Section 3.3, the proposed cloud storage service connects directly users to cloud providers. The proposed prototype allows users to store their documents while respecting the user privacy from an application that installs on the client device running a Windows 10 operating system. For a more realistic evaluation, this application does not use an infrastructure designed for this specific experiment but three of the main cloud storage providers available on the market, i.e., Dropbox, Google Drive and Microsoft OneDrive. So, users must first create at least two accounts on these cloud providers before using the application.

The application saves the documents of users on every registered cloud provider by breaking them into blocks. The number of blocks created for one document represents a multiple of the number of cloud providers to ensure an equal distribution of the data. The number of created blocks on each provider to encode one document is caculated according to the following formula:

$$N_{block} = \lceil S_{doc} / N_{provider} / S_{block} \rceil,$$

where N_{block} is the number of blocks per provider,

S_{doc} is the size of the document,

$N_{provider}$ is the number of registered providers,

S_{block} is the size of one block

From the number of providers $N_{provider}$, the encoding algorithm opens $N_{provider}$ buffers and starts to read the document content one byte at a time. Bytes are alternately distributed in each buffer as described in Figure 2. When buffer sizes reaches the block size, buffers are uploaded and $N_{provider}$ new buffers are created and filled. The process continue until the end of the document content. From this encoding algorithm, most of the blocks have an equal size and only the last $N_{provider}$ blocks are smaller than the block size. Obviously, the block size S_{block} has a huge impact on the number of blocks.

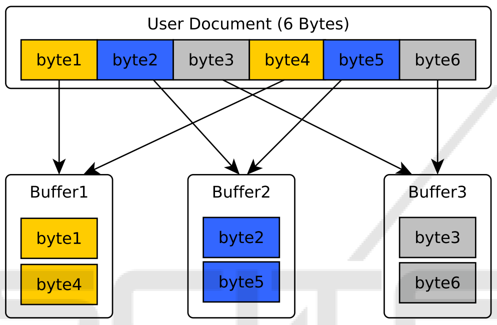
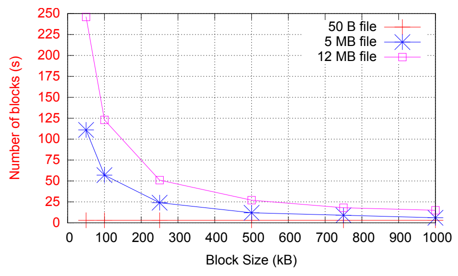


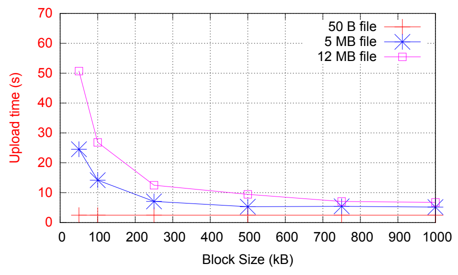
Figure 2: Distribution of one user document into 3 blocks.

The information required to reconstruct documents, i.e., the metadata, consists of a list of block names and a list of cloud providers on which blocks are stored. From these two pieces of information, the application manages blocks by uploading, downloading or deleting them by using the cloud provider REST API. We assume that users do not share their cloud accounts and the storage space provided by cloud providers is only used by one person. Consequently, the number of blocks generated by the user will not be mixed with blocks of other users, and so, it must be sufficiently high to protect the user privacy.

In the following experiments, three accounts of cloud providers are registered in the application (one Dropbox account, one Google Drive account and one Microsoft OneDrive account). The prototype will be compared with two cloud storage services: Dropbox and BoxCryptor. Dropbox is one of the famous cloud storage services. It allows users to easily save documents on the cloud. To upload documents, users move their documents in the Dropbox folder. This folder is automatically synchronized with the cloud folder. A local copy remains at the user side to read and edit the document. BoxCryptor allows users to save documents on the cloud while protecting their



(a) Number of blocks to encode documents with different block sizes



(b) Time to upload documents with different block sizes

Figure 3: The impact of the block size on document uploads.

privacy by using encryption. BoxCryptor connects to many cloud providers like Dropbox to save encrypted documents but the entire document is saved on a single provider. As BoxCryptor does not save user passwords, both cloud providers and BoxCryptor can not read user documents. So, users must trust BoxCryptor to not store their passwords. Moreover, the underlying cloud providers has entire documents so it can try to decrypt them.

As both Dropbox and BoxCryptor keep a local copy of user documents, users do not have to download their documents before reading or editing them. In our prototype, this local copy is deleted to minimize the consequences of theft of user devices. So, user documents are most often downloaded. As documents are not downloaded with Dropbox and BoxCryptor, we choose to compare upload times in the following experiments. The Section 4.2 shows the importance of the block size on the performance of the prototype. The Section 4.3 compares the prototype with Dropbox and BoxCryptor.

4.2 Performance Assessment

The prototype developped in this study is available from the Windows Store[3] for mobile phones and desktop computers running a Windows 10 operating

[3]https://www.microsoft.com/en-us/store/p/trustydrive/9nblggh52pq6

system. Every experiment is performed on a desktop computer with 4 GB of memory and a dual core processor (T8100 2.10GHz). Three documents are uploaded on the three registered providers:

- a text document with a small size of 50 B. This document could help users to save important passwords which are rarely used ;
- an audio file with a medium size of 5 MB. This document could be an interview about a sensitive topic ;
- a JPEG picture of 16 Mpixels with a size of 12 MB. The user does not want to disclose it.

The proposed storage service protects the user privacy by breaking user documents into blocks, and then, distributing these blocks to many cloud providers. Additionally, there is no third-party between users and cloud providers and only users know the metadata to prevent providers reading documents. In this section, we will examine the performance of the prototype by uploading the 3 documents described above with different block sizes. The first part of this evaluation shows the impact of the block size on transfer times. In a second phase, the system is used with a fixed block size and the encoding process is examined very carefully to differentiate computation times and transfer times.

The number of blocks used to encode one document is defined by the block size as described in Section 4.1. When the block size grows, the number of generated blocks per document decreases quickly toward the minimal number of 3 blocks per document, i.e., one block per provider. As 3 providers are registered to the application, the encoding process builds at least 3 blocks that it fills alternatively with document data. When blocks reach the maximum block size, 3 additional blocks are created. So, a file with a size smaller than 3 times the block size is always encoded on 3 blocks. The 12 MB file is encoded on 246 blocks with 50 kB blocks and 15 blocks with 1 MB blocks as shown in Figure 3(a). Reducing the number of blocks allows to decrease the number of upload requests. When the number of upload requests drops, the time to upload documents is improved as shown in Figure 3(b). When blocks vary in size from 50 kB to 500 kB, upload times of both the 5 MB file and the 12 MB are significantly improved because the number of connections to cloud providers decreases.

The upload of one document consists of filling blocks by reading and mixing the document content, uploading blocks and uploading the metadata. In the next experiment, we measure the time consumed by each of these steps to determine which one is the most consuming task. The first step is to read the document content and fill blocks while mixing their content. This step is a CPU-intensive task, and so, its duration is highly dependent on the hardware. The time to execute this step is called the *calculation time*. The second measurement is the upload of every block belonging to the document. This time is called the *upload time* and it does not include the metadata upload time. Metadata consists of block names used by the three documents and providers that store these blocks. So, the metadata size depends on the number of blocks determined by the block size. As the set of documents is small, the size of the metadata varies from 3 kB to 30 kB depending on the block size but the time to upload it is almost constant. Actually, this time varies from 0.8 s to 1.2 s but this change is more dependent on the network quality than the metadata size. Calculation times and upload times with different block sizes are summarized in the table 1.

For both the 5 MB file and the 12 MB file, a reduction in upload times is observed while the number of blocks abruptly decreases due to an increase of the block size. These new results confirm the previous ones. For small block sizes, calculation times are negatively impacted. When the block size is greater than 100 kB, calculation times are stable. Consequently, reducing the number of blocks improves upload times but it may compromise the user privacy, and so, a tradeoff between the performance and the security must be found, for example, by using 500 kB blocks.

Table 1: Calculation times and upload times during the encoding process.

5 MB File		
Block size (B)	Calculation time (s)	Upload time (s)
50 000	2.040	23.552
100 000	1.703	13.180
250 000	1.597	6.074
500 000	1.536	4.305
750 000	1.531	4.445
1 000 000	1.515	4.195

12 MB File		
Block size (B)	Calculation time (s)	Upload time (s)
50 000	4.854	49.678
100 000	3.809	25.837
250 000	3.463	11.496
500 000	3.328	8.438
750 000	3.344	6.055
1 000 000	3.219	5.758

Table 2: Upload times (s) on different cloud storage services.

File Size (MB)	20	50	100
Dropbox	14.0	26.8	65.2
BoxCryptor	18.1	30.3	65.5
Our Prototype	14.9	25.4	59.7

4.3 Storage Service Comparison

Many cloud storage services are available to store gigabytes of data like Dropbox, Google Drive, Microsoft OneDrive. These services provide a free and large storage space for users without ensuring the user privacy. Few storage services care about privacy with zero knowledge cloud solutions based on encryption like SpiderOak or BoxCryptor. To compare the performance of our prototype with other storage services, we choose two cloud services: Dropbox and BoxCryptor. Dropbox is one of the leader in storage services and BoxCryptor proposes free accounts with AES-256 encryption.

For this experiment, three files are tranferred: one 20 MB file, one 50 MB file and one 100 MB file. The prototype is configured to use 500 kB blocks. The upload times of these files are shown in the Table 2.

The results show that our prototype initially appears a little more performant but BoxCryptor and Dropbox begin by copying user documents on the local storage before uploading them. Our prototype do not use local copies. However, our prototype uploads documents to multiple cloud providers by means of concurrent block uploads that could speed up the file transfer. According to network disturbances and the accuracy of the measurement, the difference between the upload times are not sufficient to determine the fastest cloud storage service.

5 CONCLUSION

In this paper, we propose a cloud storage service that protects the privacy of users by breaking user documents into blocks in order to spread them on several cloud providers. As cloud providers only own a part of the blocks and they do not know the block organization, they can not read user documents. Moreover, the storage service connects directly users and cloud providers without using a third-party as is generally the practice in cloud storage services. Consequently, users do not give critical information (security keys, passwords, etc.) to a third-party.

To improve the presented prototype, the ability of sharing documents between users should be available. As every cloud service has this feature, our prototype does not have to copy blocks from cloud providers of the first user to providers of the second user. It just has to share the metadata of the file, and then, download the blocks from sharing links.

REFERENCES

Abu-Libdeh, H., Princehouse, L., and Weatherspoon, H. (2010). Racs: A case for cloud storage diversity. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, pages 229–240, New York, NY, USA. ACM.

Cincilla, P., Boudguiga, A., Hadji, M., and Kaiser, A. (2015). Light blind: Why encrypt if you can share? In *SECRYPT 2015 - Proceedings of the 12th International Conference on Security and Cryptography, Colmar, Alsace, France, 20-22 July, 2015.*, pages 361–368.

Dingledine, R., Freedman, M. J., and Molnar, D. (2000). The free haven project: Distributed anonymous storage service. In *In Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, pages 67–95.

Genkin, D., Shamir, A., and Tromer, E. (2014). *RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis*, pages 444–461. Springer Berlin Heidelberg, Berlin, Heidelberg.

Luo, J., Xu, L., and Plank, J. (2009). An efficient xor-scheduling algorithm for erasure codes encoding. In *Dependable Systems Networks, 2009. DSN '09. IEEE/IFIP International Conference on*, pages 504–513.

Pottier, R. and Menaud, J. M. (2016). Trustydrive: a multi-cloud storage service that protects your privacy. In *2016 IEEE 9th International Conference on Cloud Computing*.

Rabin, M. O. (1989). Efficient dispersal of information for security, load balancing, and fault tolerance. *J. ACM*, 36(2):335–348.

Singh, Y., Kandah, F., and Zhang, W. (2011). A secured cost-effective multi-cloud storage in cloud computing. In *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pages 619–624.

Waldman, M. and Mazieres, D. (2001). Tangler: A censorship-resistant publishing system based on document entanglements. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, CCS '01, pages 126–135, New York, NY, USA. ACM.