Sharing Files Using Cloud Storage Services

Tiago Oliveira, Ricardo Mendes, and Alysson Bessani
{toliveira,rmendes}@lasige.di.fc.ul.pt, bessani@di.fc.ul.pt

Universidade de Lisboa, Faculdade de Ciências, LaSIGE – Portugal

Abstract. The widespread use of cloud storage in the last few years can be attributed to the existence of appealing applications such as file backup, data archival and file sharing. File sharing in particular, is implemented in different ways by distinct cloud storage services. These differences can appear at the offered permission types and in the form they are applied. We present a survey of these differences for several popular cloud storage services. We also show how to realize secure data sharing using these services, allowing the implementation of equivalent data sharing features in different clouds, an important requirement for secure multi-cloud systems.

1 Introduction

With more people accessing their files online, an important part of file sharing today is done by taking advantage of cloud storage. This can be done through personal file synchronization services like Dropbox [4], Google Drive [6], Microsoft OneDrive [13], Box [3] or Ubuntu One [18], which store users' data in the cloud. These services have been extremely successful, as attested by the success of DropBox, which has announced last April that it reached 275 million users [5].

These systems perform file sharing through dedicated application servers which are responsible for controlling access to the files as well as user groups management, data deduplication, etc. It means that the security of the file sharing requires trusting not only the storage service (for instance, Dropbox is built on top Amazon S3 [4]), but also these application servers.

An alternative for using these services is to mount the cloud storage (e.g., Amazon S3) in a user-level file system and access it directly. S3QL [17], BlueSky [26] and SCFS [24] are examples of this kind of systems.

BlueSky uses a proxy that acts as a network file server, which is accessed by the clients in order to store their data. This proxy is responsible for sending the users' data to the storage clouds. Nonetheless, as in synchronization services, clients need to trust this component and the cloud storage provider.

On the other hand, S3QL and SCFS allow clients to share data without a proxy. In S3QL, the clients just mount the file system to access the storage service objects as files, with no concurrency control. SCFS, on the other hand, offers controlled file sharing where concurrent updates and file version conflicts are avoided through the use of locks. Moreover, in SCFS clients are able to take advantage of DepSky [23] to store data in a multiple cloud providers, i.e., a *cloud-of-clouds*. DepSky and SCFS ensure the privacy, integrity and availability

of the data stored in the clouds as long as less then a third of the cloud providers are faulty.

SCFS uses the *pay-per-ownership* model, in which each user pays for the files he/she creates. A more simple model where all the clients use the same cloud account could be used, and automatically share all the data stored by the system. This alternative raises some problems. First, all users could access all data stored in the clouds. In this way, each client must trust all the system users since they can access, delete or corrupt all stored data. Second, just one organization will be charged for all the data stored in the system.

In this paper we present a survey of the access-control techniques provided by some popular cloud storage services. We also show how to implement secure data sharing using these services, allowing the implementation of equivalent data sharing features in different clouds, a fundamental requirement for multi-cloud systems.

In summary, we contribute with (1) a study of several cloud storage services' access control models (i.e., Amazon S3 [1], Google Storage [7], Window Azure Storage [19], RackSpace Cloud Files [15], HP Public Cloud [9], and Luna Cloud [11]), i.e, a study of the techniques used by these services to apply the permissions they provided and (2) a set of protocols that allow the sharing of files between clients according with pre-defined access control patterns for each of the studied storage clouds.

2 Access control on storage clouds

To allow users to share their data, all cloud storage services provide some mechanisms that enable data owners (users) to grant access over their resources to other principals.

In all these storage services, the resources can be either *buckets* or *objects*. A bucket, or *container*, represents a root directory where objects must be stored. There could be several buckets associated with a single cloud storage account.[1] However, in most of the services, buckets must have unique names. Objects are stored in a bucket and can be either files or directories.

On the other hand, the cloud storage services differ in the techniques they provide to allow users to grant access over their resources, and also in the permission types that users are able to specify. In this paper these techniques will be called *access-granting* and the permission types that can be specified with them will be named *permissions*.

Access-granting techniques. These are the techniques provided by cloud storage services to allow users to give others access to their resources. The users are able to specify a set of permissions in each technique. To apply these permissions over the resources, different storage clouds could offer different techniques. In this paper we cover three of them: *Per Group Predefined Permissions*, *Temporary Constraints* and *Access Control Lists* (ACLs).

[1] Some storage clouds has a limited number of buckets. For instance, an Amazon S3 account can have at most 100 buckets.

In the first one the users are able to make their stored data accessible to some predefined group. The second technique allows users to give other users a ticket that grant access for a resource by a predefined period of time. The last one, ACLs, permits to associate with each resource a list of grantees that are able to access it.

Permissions. When a user wants to share some resource with another user, i.e., with a different account/user, he/she needs to specify what are the capabilities of this other user with respect to the shared resource(s). Permissions are specified in the access-granting techniques. Each permission has a semantic that specifies the capabilities of the grantees of some resource.

However, different storage clouds provide a different set of specifiable permissions. For example, Amazon S3's users cannot grant WRITE permission to specific objects. On the other hand, RackSpace Cloud Files users can do that [15] (see Section 3).

Moreover, equal permissions could have different semantics. As an example, in Amazon S3 [1] when a READ permission over a bucket is given, grantees can list objects inside it. On the other hand, the same permission in Windows Azure [19] does not allow grantees to list the objects in a bucket, instead it grants the permission to read all the objects it contains.

3 Permissions

To allow users to define the grantees' capabilities over a shared resource, all clouds provide a set of permissions with documented semantics. As explained before, the same permission could have different semantics in different clouds. Table 1 shows the available permissions for buckets and objects in several cloud storage providers.

As can be seen, Amazon S3 [1] and LunaCloud [11] provide the largest set of permissions among all the services studied. They permit users to give READ, WRITE, READ_ACP, WRITE_ACP and FULL_CONTROL permissions [2, 12] over both buckets and objects. Google Storage [7] have almost the same set of permissions. The difference is that Google Storage does not allow users to apply READ_ACP and WRITE_ACP permissions separately [8]. Instead, it put together these two permissions into the FULL_CONTROL one. This means that if a user wants to give other users the capability of read some resource's ACL, he/she is forced to also grant the capability to write or update that ACL.

Interestingly, in most clouds the READ permission over a bucket does not allow a grantee to read an object inside it. Instead, it only allows grantees to list the objects inside the bucket. To grant read access, a READ permission need to be applied on the desired objects. On the other hand, the WRITE permission on the bucket allows a grantee to write, overwrite or delete any object inside that bucket. In this case, the same permission is not applicable to objects. Given that, it is impossible to grant WRITE access to a subset of the objects inside a bucket. This means that there is no way to grant write access over objects individually.

Another important thing to highlight is that those clouds allow users to give others the right to read or write a resource's ACL through the READ_ACP,

	Available Permissions	
	On bucket:	**On object:**
Amazon S3	•READ: List the objects in the bucket. •WRITE: Create, overwrite, and delete any object in the bucket. •READ_ACP: Read the bucket ACL. •WRITE_ACP: Write the ACL for the applicable bucket. •FULL_CONTROL: READ, WRITE, READ_ACP, and WRITE_ACP permissions on the bucket.	•READ: Read the object data and its metadata. •WRITE: Not applicable. •READ_ACP: Read the object ACL. •WRITE_ACP: Write the ACL for the applicable object. •FULL_CONTROL: READ, READ_ACP, and WRITE_ACP permissions on the object.
Google Storage	•READ: List a bucket's contents. •WRITE: List, create, overwrite, and delete objects in a bucket. •FULL_CONTROL: READ and WRITE permissions on the bucket. It also lets a user READ and WRITE bucket ACLs and other metadata.	•READ: Download an object. •WRITE: Not applied. •FULL_CONTROLL: READ access. It also lets a user READ and WRITE object ACLs and other metadata.
HP Public Cloud	•READ: Read and list any object in the bucket. •WRITE: Create, overwrite and delete any object in the bucket.	•READ: Read the specified object. •WRITE: Write in the object.
RackSpace	•READ: Read any object in the bucket. •WRITE: Create, overwrite and delete any object in the bucket.	•READ: Read the specified object. •WRITE: Write in the object.
Windows Azure	•READ: Read any object in the bucket. •WRITE: Write and overwrite any object in the bucket. •DELETE: Delete any object in the bucket. •LIST: List the objects in the bucket.	•READ: Read the specified object. •WRITE: Write the specified object. •DELETE: Delete the specified object.
LunaCloud	•READ: List the objects in the bucket. •WRITE: Create, overwrite, and delete any object in the bucket. •READ_ACP: Read the bucket ACL. •WRITE_ACP: Write the ACL for the applicable bucket. •FULL_CONTROL: READ, WRITE, READ_ACP, and WRITE_ACP permissions on the bucket.	•READ: Read the object data and its metadata. •WRITE: Not applicable. •READ_ACP: Read the object ACL. •WRITE_ACP: Write the ACL for the applicable object. •FULL_CONTROL: READ, READ_ACP, and WRITE_ACP permissions on the object.

Table 1. Storage services available permissions.

WRITE_ACP and FULL_CONTROL permissions. It is also important to notice that when a grantee have the permission to update an ACL, he/she is able to grant access over it to other users without being the resource owner.

HP Public Cloud [9] and RackSpace Cloud Files [15] available permissions are more simple [10, 14]. These two services only provide two different permissions, either for buckets and objects: READ and WRITE. The only difference between these two storage clouds is that the READ permission on the bucket for Hp Public cloud allow grantees to list and read its objects (contrary to Amazon S3, LunaCloud and Google Storage), while for RackSpace only allow grantees to read the objects inside it. Also different of Amazon S3, LunaCloud and Google Storage, in these two clouds is impossible give other users the right to read/update the bucket permissions.

The Windows Azure Storage's [19] set of permissions [20] differs from all other studied cloud storage services. Basically, the WRITE and DELETE permissions are separated, as well as the READ and the LIST. In the other clouds these permissions are grouped in one permission, i.e., when the WRITE access is granted, grantees have also the capability to delete resources. Similarly with HP Public Cloud and RackSpace, a grantee cannot update/read the bucket permissions.

4 Access-granting Techniques

4.1 Per Group Predefined Permissions

Using Per Group Predefined Permissions, the users are able to apply permissions on buckets or objects granting access for two kinds of groups: *All Users*

and *Authenticated Users*. The All Users group refers to anyone in the internet. In turn, the Authenticated Users group represents all users that have an account in the cloud provider. However, to give permissions to these groups, the owners must use some predefined permissions that the storage clouds provide. For instance, Amazon S3 and LunaCloud call this technique *Canned ACLs*, while Google Storage name it *Predefined ACLs*.

The first column of Table 2 shows the available predefined permissions for each group, as well as the type of access that each one grants. As we can see, Amazon S3 and Google Storage provide the same Per Group Predefined Permissions [2, 8].[2] These default permissions allow users to make their resources public for the All Users group, for both READ and WRITE. For the Authenticated group, the storage clouds only allow the users to grant READ permission over buckets or objects. Notice that the *bucket-owner-read* and *bucket-owner-full-control* predefined permissions over objects for Amazon S3 and Google Storage, only grant access permissions to resources owners (not for the Authenticated Group). These predefined permissions are provided because the bucket owner could not be the object owner. In these two clouds, each user is the owner of the objects he uploads, even if the uploads are made to a bucket owned by other user. Thus, they are useful to give access rights to the bucket owner when a user uploads an object to a bucket that is not owned by him. The LunaCloud's predefined permissions are similar to the Amazon S3 and Google Storage (see Table 2), with the exception that they do not have the *bucket-owner-read* and *bucket-owner-full-control* permissions over the objects.

Windows Azure differs from the previous clouds in two ways. First, there is no predefined permissions to the Authenticated Users group. Second, it provides no way to give WRITE permissions over buckets or objects, allowing only users to grant READ access to the All Users group [21].

Although not shown in the table, all the clouds that provide predefined permissions also provide a special permission that gives FULL_CONTROL to the bucket/object owner, with no one else getting any access to it. In fact, this is the default predefined permission for a resource on its creation.

HP Public Cloud and RackSpace Cloud Files do not provide Per Group Predefined Permissions. However they allow users to make their buckets public through different techniques.

4.2 Temporary Constraints

Temporary Constraints is another way to give access permissions to other users. However, using this technique, the access will be temporary. The second column of Table 2 shows the studied clouds that implement this access-granting technique. As can be seen, only three of the studied clouds have this feature. RackSpace and HP Public Cloud provide *Temporary URLs* [16, 10], while Windows Azure provide *Shared Access Signatures* [22]. Temporary URLs are used to

[2] There are some other predefined permissions for this two storage clouds that are not shown in the figure.

support the sharing of objects (and only objects), whereas Shared Access Signature allows the sharing of both buckets and objects. These temporary constraints work as a capability given by resource owners to other users in order for them to access to the specified resource. In this case the ticket that proves the right to access the object is the URL. This URL contains information about the period of time that the access will be valid, the path to the resource over which the access is being granted, the permissions granted, and a signature. This signature, not to be confused with a digital signature [25], is different from cloud to cloud:

	Per Group Predefined Permissions		**Temporary Constraints**	ACLs
	All Users	**Authenticated Users**		
Amazon S3	•public-read (bucket and object): Owner gets FULL_CONTROL. The All Users group gets READ access. •public-read-write (bucket and object): Owner gets FULL_CONTROL. The All Users group gets READ and WRITE access.	•authenticated-read (bucket and object): Owner gets FULL_CONTROL. The Authenticated Users group gets READ access. •bucket-owner-read (object): Object owner gets FULL_CONTROL. Bucket owner gets READ access. •buclet-owner-full-control (object): Both the object owner and the bucket owner get FULL_CONTROL over the object.	✖	✔
Google Storage	•public-read (bucket and object): Owner gets FULL_CONTROL. The All Users group gets READ access. •public-read-write (bucket and object): Owner gets FULL_CONTROL. The All Users group gets READ and WRITE access.	•authenticated-read (bucket and object): Owner gets FULL_CONTROL. The Authenticated Users group gets READ access. •bucket-owner-read (object): Object owner gets FULL_CONTROL. Bucket owner gets READ access. •buclet-owner-full-control (object): Both the object owner and the bucket owner get FULL_CONTROL over the object.	✖	✔
HP Public Cloud	✖	✖	TempURL	✔
RackSpace	✖	✖	TempURL	✔
Windows Azure	•full-public-access: All Users group gets READ and LIST access. •public-read-access-for-blobs-only: All Users gets READ access .	✖	Shared Access Signature	✖
LunaCloud	•public-read (bucket and object): Owner gets FULL_CONTROL. The All Users group gets READ access. •public-read-write (bucket and object): Owner gets FULL_CONTROL. The All Users group gets READ and WRITE access.	•authenticated-read (bucket and object): Owner gets FULL_CONTROL. The Authenticated Users group gets READ access.	✖	✖

Table 2. Storage services access-granting techniques.

- RackSpace Cloud Files: SHA-1 HMAC computed over the URL information and a key.
- HP Public Cloud: SHA-1 HMAC computed over the URL information and a key.
- Windows Azure Storage: SHA-256 HMAC computed over the URL information and a key.

In the first case, the key is a sequence of letters chosen by the user, while in the others, it is the secret key used to access the account. This signature ensures (with high probability) that the URL cannot be guessed or changed by an attacker even if he knows the other fields of the URL.

4.3 Access Control Lists - ACLs

As shown in Table 2, Amazon S3 [2], Google Storage [8], HP Public Cloud [10] and RackSpace Cloud Files [14] are the clouds that allow users to specify access rights to other users through ACLs. Contrary to the use of Temporary Constraints, by using ACLs, the user does not need to give to grantees a capability (a URL like described in Section 4.2). In this case the user who wants to share data needs to create an ACL and include the names or ids (depending on the cloud) of the clients whom he want to give access together with the corresponding permissions and associate it with the shared objects.

However, there are some differences among the storage clouds that provide ACLs. One difference between Amazon S3 and Google Storage, and RackSpace and HP Public Cloud is that in the last two, the users can only manage ACLs for containers. This means that it is impossible for a user to associate an ACL with an object. Another difference is that, while Amazon S3 and Google Storage allow users to grant access to a user from a different account, RackSpace and HP Public Cloud only allow they to set an ACL for sub-users.[3]

All clouds that allow sharing across different accounts through ACLs, do not permit buckets to have the same name, even if they belong to different accounts.

5 Setting Per User Permissions

Table 3 summarizes which clouds implement mechanisms for securely sharing buckets and objects between different users (which is a requirement for implementing the cloud-of-clouds models of DepSky [23] and SCFS [24]). Among the studied clouds, LunaCloud is the only one that does not provide enough features for this, since it only provides Per Group Predefined Permissions. In the remaining clouds, the per user sharing can be done through ACLs or Temporary Constraints. However, none of these clouds provide mechanisms for securely sharing a bucket in a simple way.

To clarify what we mean by "securely" and "simple", we define a minimum set of rules to share a bucket in a secure way:

- ***Rule A***: the permissions on the bucket allow a grantee to list, delete, create, read and write any object inside it.
- ***Rule B***: only grantees and the bucket owner can operate on the bucket.
- ***Rule C***: a grantee cannot delegate access rights to other users.

Sharing a container using a cloud storage service that satisfies these rules and support ACL as access-granting technique would be very simple. First, the bucket owner gathers the ids of the accounts he wants to grant access to, and then he creates/associates a bucket with an ACL granting the desired permissions for those accounts.

Unfortunately, as described before, this simple protocol cannot be applied to any cloud we are aware of. However, equivalent functionalities can be implemented in most clouds, albeit using additional steps. In the following subsections

[3] A sub-user is a user within an account owned by other user. Such users can be associated to an account by associating with him a username and a password.

we present the steps required for sharing a bucket with specific users in the different clouds in which this is possible.

		Amazon S3	Google Storage	HP Public Cloud	RackSpace	Windows Azure	LunaCloud
Per User Permission	on bucket:	ACL	ACL	ACL	ACL	Temp. Constraints	✖
	on object:	ACL	ACL	Temp. Constraints	Temp. Constraints	Temp. Constraints	✖

Table 3. Per user permissions in storage services.

5.1 Sharing with Amazon S3 and Google Storage

Sharing a bucket among specific users in Google Storage and Amazon S3 is quite similar. In the following we describe a protocol (illustrated in Figure 1) for sharing a bucket in these storage clouds.

1. The bucket owner needs to gather the ids of the users he want to share with. In the case of Amazon S3, this is the *Canonical User ID* while for Google Storage this is the email associated with the account.
2. The bucket owner must create the bucket and associate with it an ACL with the ids of grantee X and Y granting READ and WRITE permissions. The bucket owner can always get the bucket ACL from the cloud, add more or remove users to it, and update it again.
3. All the ids, including the id of the bucket owner, must be sent to all grantees.
4. When a grantee or the bucket owner uploads an object, it needs to associate an ACL granting READ access to the other grantees (including the bucket owner).

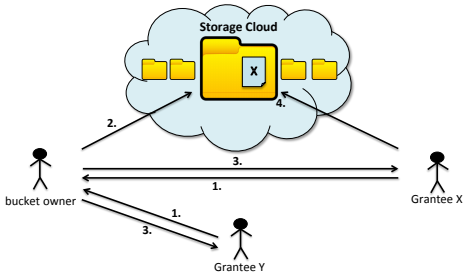

Fig. 1. Sharing a bucket with Amazon S3 and Google Storage.

The permissions provided by these two clouds are different from the set of permissions we defined in Section 5, therefore, this protocol is more costly than the protocol exemplified in that section. For instance, *Rule A* is not respected in the second step: there is no permission on the bucket that allows grantees to read all objects inside it. This leads to the third and fourth step described above: when an object is uploaded to the shared bucket we need to associate an ACL with it to ensure READ access to all grantees.

Assuming that all grantees are trusted, this algorithm fulfill all the requirements present in the set of rules we defined in Section 5. However, if any of them deviate from the protocol, some new issues can arise. In these clouds each user is the owner of the objects he uploads, consequently, he/she can grant READ access over his objects to other users without the knowledge of bucket owner, or even give no access to other grantees. The first case does not respect *Rule C*, while the last case contradicts *Rule A*. Notice that it is impossible to the grantee to give write access to others because these two clouds provide no WRITE permission for objects. If the bucket owner detect these situations, it can always delete the objects the grantee uploaded and revoke his access permissions.

5.2 Sharing with HP Public Cloud and RackSpace Cloud Files

HP Public Cloud and RackSpace are the only two clouds, of the five we studied, that allow per user permissions and that provide ACLs and Temporary Constraints to share resources. However, their temporary URLs only allow users to share objects, not buckets (see Section 4.2). Figure 2 illustrates the steps required for sharing a container using ACLs.

1. The bucket owner needs to get the grantees' names and emails. This is the information needed to add a sub-user. Notice that the grantees do not need to have an HP Public Cloud or RackSpace account.
2. The bucket owner adds the grantees as sub-users of its account. By default, a sub-user cannot access any service until the account owner allows it.
3. After that, the bucket should be created and an ACL with READ and WRITE permissions granting access for the previous added users must be associated with it. For RackSpace in particular, there is no way to update an ACL already associated with a bucket in the cloud, only to replace it. This means that if the bucket owner updates an ACL only granting access to grantee X, when updating the ACL for giving access to grantee Y, the access to grantee X must be granted again.
4. The next step is to provide to grantees the credentials they need to get authenticated as sub-users. The bucket owner can get these credentials after adding the grantees to its account (step 2).
5. From now on, the grantees can authenticate themselves with cloud service using the referred credentials.
6. Thereafter, they can operate in the bucket that was granted access.

Since RackSpace do not allow grant list access to grantees (see Section 3), in this case *Rule A* is not respected. However, in the case of HP Public Cloud all rules that we specify in Section 5 are covered. Despite that, in both cases *Rule B* and *Rule C* are respected as long as all grantees are trusted. Otherwise, a non-trusted grantee can provide non-authorized users with the access credentials, making them able to read, write, delete and list on the bucket. This contradicts *Rule C*. However, the bucket owner can revoke grantees permissions just by deleting them from the bucket ACL, or even by removing their sub-users.

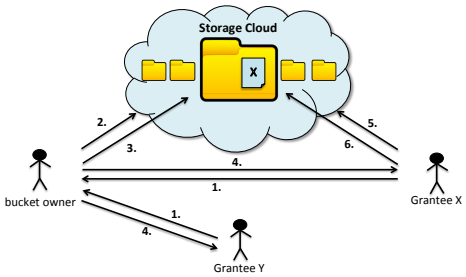

Fig. 2. Sharing a bucket with HP Public Cloud and RackSpace Cloud Files.

Another issue is that these two clouds do not allow sharing across different accounts. This increase the number of necessary steps of the protocol. More specifically, there is the need of steps 2, 4 and 5. It is important that the step 4 is executed over a secure connection to ensure that no one can read the access credentials from the network.

5.3 Sharing with Windows Azure

The only way to share a container with other users using Windows Azure is through a Temporary Constraint. Figure 3 illustrates the required steps.

1. The bucket owner creates the bucket he wants to share.
2. Generate the Shared Access Signature to this bucket with all the permissions that Windows Azure provide: READ, WRITE, DELETE and LIST.
3. Disseminate the URL among the grantees.
4. Once the grantee have the URL, he can use it to access the bucket for READ, WRITE, DELETE and LIST until its expiration.

As in the previous protocols, there is the need to assume that the grantees are trusted. The second step of the protocol allows the creation of a URL with the rules we define in Section 5. However, using a Temporary Constraint as an access-granting technique raises some issues. Firstly, step 3 should be done via a secure connection. The second one relates to the fact that a grantee can give to others the URL, thus breaking *rule C*. This means that other users can access the bucket for read, write, list and delete. Finally, there is the inconvenience of repeating the process (with the exception of step 1) every time the URL expires.

5.4 Suggestions for improvements

In each cloud service there are some aspects that can be modified in order to make secure sharing easier. Here they are summarized.

Amazon S3 and Google Storage. As explained above, when a user wants to give the capability of read the content of the files inside a bucket, it needs to give the READ permission to each object inside that bucket. This obviously does not scale in applications with a large number of objects. To solve this problem,

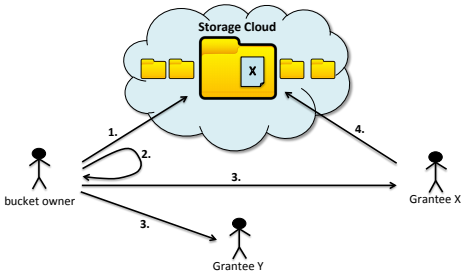

Fig. 3. Sharing a bucket with Windows Azure.

these services should provide bucket permissions that give users read access to all objects inside that bucket. In order to prevent grantees to give others access to files they upload, these clouds just need to use a model where the bucket owner is the owner of all objects inside the buckets it pays for, instead of the model where the owner of an object is the user who uploads it.

RackSpace Cloud Files and HP Public Cloud. These clouds require additional steps for defining the credentials for grantees to have access to the shared data. These steps are needed because neither of these services allow cross-account sharing, and could be avoided if this was supported. Specially for RackSpace Cloud Files, a permission to grant list access to grantees should be provided.

Windows Azure Storage. With Shared Access Signatures a malicious grantee is always able to give others the URL allowing anyone to access the shared resources. To solve this issue, Windows Azure just need to provide other access-granting technique, such as ACLs, allowing users to share data between accounts.

6 Conclusion

This paper presents a study of the access control capabilities of some storage cloud services that permit users to share data using the unmodified clouds directly and assuming a model where each user pays for the storage of the objects he/she creates. The storage clouds studied were Amazon S3, Google Storage, HP Public Cloud, RackSpace Cloud Files, Windows Azure Storage and Luna Cloud.

We described the permissions provided by the services, their semantics, and the different access-granting techniques that are used to apply these permissions to specific users. Additionally, a set of protocols for sharing data securely in several public storage clouds were presented. These protocols were defined by extending an ideal set of properties required for sharing data between different users of a cloud service.

We concluded that none of the studied cloud services offer the tools to implement an optimal solution that respect all these properties, but it is possible to implement sharing in most of them.

Acknowledgments. This work is supported by EC's and FCT through projects BiobankCloud (FP7-317871) and LaSIGE (PEst-OE/EEI/UI0408/2014).

References

1. Amazon s3. http://aws.amazon.com/s3/
2. Amazon s3 documentation. http://docs.aws.amazon.com/AmazonS3/latest/dev/ACLOverview.html
3. Box. https://www.box.com/
4. Dropbox. https://www.dropbox.com/
5. Dropbox number of users announcement. http://techcrunch.com/2014/04/09/dropbox-hits-275m-users-and-launches-business-product-to-all/
6. Google drive. https://drive.google.com/
7. Google storage. https://developers.google.com/storage/
8. Google storage documentation. https://developers.google.com/storage/docs/accesscontrol
9. HP public cloud. http://www.hpcloud.com/products-services/storage-cdn
10. HP public cloud documentation. https://docs.hpcloud.com/api/object-storage#general_acls-jumplink-span
11. Lunacloud. http://www.lunacloud.com/pt/cloud-storage
12. Lunacloud predefined permissions documentation. http://www.lunacloud.com/docs/tech/storage-restful-api.pdf
13. Microsoft onedrive. https://onedrive.live.com/about/pt-br/
14. Rackspace acls documentation. http://www.rackspace.com/blog/create-cloud-files-container-level-access-control-policies/
15. Rackspace cloud files. http://www.rackspace.co.uk/cloud/files
16. Rackspace temporary urls documentation. http://docs.rackspace.com/files/api/v1/cf-devguide/content/TempURL-d1a4450.htmll
17. S3QL - a full-featured file system for online data storage. http://code.google.com/p/s3ql/
18. Ubuntu one. https://one.ubuntu.com/
19. Windows azure. http://www.windowsazure.com/pt-br/solutions/storage-backup-recovery/
20. Windows azure permissions documentation. http://msdn.microsoft.com/en-us/library/windowsazure/dn140255.aspx
21. Windows azure predefined permissions documentation. http://msdn.microsoft.com/en-us/library/windowsazure/dd179354.aspx
22. Windows azure shared access signature documentation. http://azure.microsoft.com/en-us/documentation/articles/storage-dotnet-shared-access-signature-part-1/
23. Bessani, A., Correia, M., Quaresma, B., Andre, F., Sousa, P.: DepSky: Dependable and secure storage in cloud-of-clouds. ACM Transactions on Storage 9(4) (2013)
24. Bessani, A., Mendes, R., Oliveira, T., Neves, N., Correia, M., Pasin, M., Verissimo, P.: SCFS: a shared cloud-backed file system. In: Proceedings of the 2014 USENIX Annual Technical Conference – ATC'14 (2014)
25. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 21(2) (1978)
26. Vrable, M., Savage, S., Voelker, G.M.: BlueSky: A cloud-backed file system for the enterprise. In: Proceedings of the 10th USENIX Conference on File and Storage Technologies – FAST'12 (2012)