Towards a Framework for Privacy-Preserving Data Sharing in Portable Clouds

Clemens Zeidler and Muhammad Rizwan Asghar

Department of Computer Science, The University of Auckland, Auckland 1142, New Zealand,
{clemens.zeidler, r.asghar}@auckland.ac.nz

Abstract. Cloud storage is a cheap and reliable solution for users to share data with their contacts. However, the lack of standardisation and migration tools makes it difficult for users to migrate to another Cloud Service Provider (CSP) without losing contacts, thus resulting in a vendor lock-in problem. In this work, we aim at providing a generic framework, named *PortableCloud*, that is flexible enough to enable users to migrate seamlessly to a different CSP keeping all their data and contacts. To preserve the privacy of users, the data in the portable cloud is concealed from the CSP by employing encryption techniques. Moreover, we introduce a migration agent that assists users in automatically finding a suitable CSP that can satisfy their needs.

Keywords: Portable cloud, privacy, data sharing, data migration, migration costs, migration agent

1 Introduction

Cloud storage is a cheap and reliable alternative to a local storage system. A Cloud Service Provider (CSP) is considered to ensure availability of cloud services so that users can get access to their data from anywhere at any time. Leveraging cloud storage can be an attractive business model for individuals as well as for enterprises that do not have resources to deploy and maintain custom storage solutions. However, data in the cloud is stored at geographically dispersed locations, thus raising serious privacy concerns. Assuming that the CSP is *honest-but-curious* [De Capitani di Vimercati et al., 2008], data has to be kept confidential. Technically, the confidentiality of the data can be guaranteed by employing encryption before storing the data in the cloud.

Many CSPs allow their users to share data with each other, which is a great way to collaborate with third parties. For example, Dropbox[1] enables users to share files with each other. However, sharing data with users who do not belong to the same CSP is usually limited and less secure and requires a manual token or key exchange with third parties. Throughout this chapter, we call third parties *contacts*, *i.e.,* the parties with whom users shares their data.

There are various reasons why a user may want to migrate her data from one CSP to another one. For example, if there are cheaper CSPs available, the service conditions

[1] https://www.dropbox.com/

have changed or the current service is not reliable enough. Furthermore, there are jurisdictional restrictions on the CSP [Joint et al., 2009]. In the worst case, a CSP might have to shut down its services for financial or legal issues. For instance, if a CSP is used for illegal file sharing, the CSP may face legal issues and its service may get interrupted. For innocent users, this can lead to loss of their personal data.

Having contacts at a certain CSP can be a hindrance for a user to migrate to another CSP since these contacts would then be lost, *i.e.,* a user and a contact would not be able to access and share data with each other anymore. Another problem that makes it hard to migrate a cloud service is that there is often a lack of tools for a seamless migration. For example, there is no simple way to migrate data between CSPs when data needs to be transformed to a different format or encryption scheme. These problems that could stop users from migrating to a different CSP are also known as vendor lock-in [Armbrust et al., 2010, De Chaves et al., 2011, Satzger et al., 2013].

In this chapter, we propose *PortableCloud*, a generic framework that addresses the problem of vendor lock-ins and allows users to seamlessly migrate data between CSPs that run *PortableCloud*. If required, data can even be removed from the cloud and migrated to a local service that runs *PortableCloud*. Data can be shared between contacts that reside either at the same or a different CSP (see Figure 1). To preserve the privacy of users, data is stored encrypted and can only be accessed by authorised parties. When migrating the portable cloud to a new CSP, all contacts are kept and automatically notified about the migration, *i.e.,* the migration is transparent to users and their contacts. We provide a migration cost analysis of the portable cloud migration. Further, we propose an agent that informs users about CSPs with better conditions in order to help them to migrate to a new CSP.

Our contributions can be summarised as follows:

- A proposal of a novel privacy-preserving portable cloud framework *PortableCloud* that enables seamless migration to a new CSP while maintaining all existing contacts.
- A cost analysis of a portable cloud migration.
- A migration agent that assists users in migrating to another CSP.

Section 2 motivates and defines requirements for *PortableCloud*. Section 3 provides an overview of the system model. Section 4 elaborates *PortableCloud*. Section 5 explains the migration process, analyses the migration cost and describes a migration agent. Section 6 discusses privacy aspects of the portable cloud and how the portable cloud can be used by enterprises. Section 7 reviews related work. Section 8 concludes this chapter and gives directions for future work.

2 Scenario and Requirements

2.1 Motivating Scenario

In this section, we briefly explain a scenario that motivates why we need a portable cloud. Let us assume an organisation that has to store data of its users who could share data with their contacts that may or may not belong to the same organisation. A typical example of such an organisation is a university, where data is shared between researchers from the same university as well as with collaborating researchers from other universities and industrial partners.

We consider that the organisation outsources storage services to a CSP. Like a typical storage system in the trusted environment, accounts are created and access rights are defined for users and external contacts for sharing data among them residing on the CSP servers. Since a CSP could compromise the privacy of users when getting cleartext access to the data, the organisation employs encryption techniques before storing any data in the cloud. Consequently, sharing data becomes a matter of sharing secret keys for the encrypted data.

The organisation might decide to migrate from an existing CSP to a new one. There could be various reasons behind such a migration. The most prominent reasons include cost, limitation of (hardware and software) resources and trust. For instance, there is a CSP that (i) offers cheaper services, (ii) does not have sufficient hardware to offer more storage or a particular version of software is not supported or (iii) a CSP is not trusted anymore due to some unexpected event or bad experience.

For migrating from one CSP to another, the organisation could face a number of problems. First, it is not easy to just copy data from one system to another. For instance, a migration process could require downloading a part of encrypted data locally and re-encryption before uploading it to the new CSP. Unfortunately, current CSPs do not support tools for aforementioned operations. Second, all users and their contacts need to be notified about the new CSP. This is not only a tedious task but could also be erroneous and can lead to serious access right issues for users in the system. Third, the new encryption keys and the new tools, which are required to access the data, need to be distributed to the users and contacts. In the case of the second and third problems, any manual change by users or contacts could result in some human errors. Fourth, during the migration process, users and contacts might face issues in accessing the data or consuming services. Even if the old CSP keeps serving till completion of the migration process, there could be inconsistencies in the data if it is modified meanwhile. In other words, there could be serious issues if any data write operation is encountered during the migration process. Last but not least, organisations need to manually explore themselves CSPs that can make competitive offers. More precisely, the existing infrastructure does not take into account automatic discovery of alternative CSPs or their services.

Currently, all these problems result in vendor lock-in issues because an organisation cannot easily choose to migrate to another CSP. An organisation has to consider any migration carefully since it could affect their users as well as their users' contacts.

2.2 Requirements

From the above scenario, we can deduce a set of requirements for a portable cloud architecture. First, the option to migrate to a set of different CSPs can lead to certain security threats since it could be difficult to judge which CSPs keep data private and could be trusted. The data must not only be encrypted, but users and contacts must be able to ensure that the data can only be read by authorised entities, is not modified by unauthorised entities [Hacigümüş et al., 2004] and the origin of the data is authenticated [Asghar et al., 2012]. More precisely, the framework of portable clouds must provide mechanisms to ensure user privacy, integrity and provenance. This can be achieved by using cryptographic methods to encrypt the data to conceal private information from the CSP and sign it. Furthermore, any information that any CSP could infer from the stored or exchanged data must be as minimal as possible.

Second, the migration should be seamless for all stakeholders, in particular for users and contacts. Moreover, the downtime for a migration should be small and contacts need to be notified automatically about the migration. This makes it easy for users to migrate to a new CSP without losing connections to existing contacts, *i.e.,* contacts are able to access data at the new CSP.

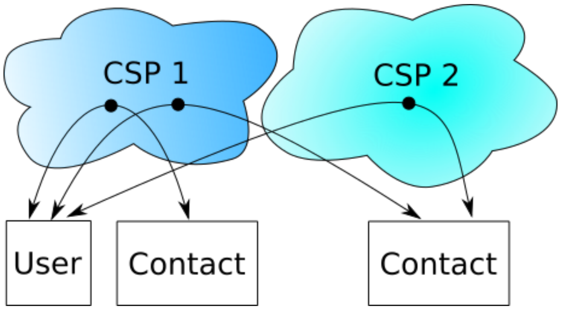

Fig. 1. Data sharing between a user and her contacts who may or may not be at the same CSP. Even after a migration to a new CSP contacts are still able to access data at the new CSP.

Third, users must be able to share data with their contacts that may or may not belong to the users' CSP (see Figure 1). This is a quite natural requirement since data sharing should stay functional after the migration.

To summarise the requirements:

1. User privacy must be preserved.
2. Data integrity and provenance must be provided.
3. A migration to a new CSP must be transparent and seamless to the user and her contacts.
4. Data needs to be shareable with contacts that may or may not belong to the users' CSP.

3 SYSTEM MODEL

A *user* of a *CSP* stores all her data at the CSP. Data at the CSP can be shared with *contacts*. Contacts are users of the same or different CSPs. In our system model, we assume the following entities:

- **Cloud Service Provider (CSP):** It is responsible for storing the data and runs the *PortableCloud* software. The CSP ensures that only authorised parties can access the data. Furthermore, the CSP manages the communication between users and their contacts.
- **User or Organisation:** It is an entity that owns the data managed by the CSP and regulates access to the data by deploying access control policies on the CSP. It is a client of the CSP. In case of an organisation, there could be an administrator. However, in case of individuals, we do not distinguish between an administrator and a user. It is responsible for the encryption of the data before storing it in the cloud.
- **Contact:** A contact is a party that can access data at the user's CSP according to an access control policy defined by the user. Contacts are users of the same or of a different CSP.

Data at the CSP could be stored in any format. It could be managed in files or there could be a database. We assume that the CSP is honest-but-curious [De Capitani di Vimercati et al., 2008]. This means the CSP runs PortableCloud correctly but the user cannot trust the CSP to provide confidentiality. For this reason, the user encrypts confidential data locally to prevent the CSP to gain cleartext access to this data. The user encrypts data using a secure symmetric key encryption algorithm, such as AES.

To be able to securely communicate with contacts, each user has a set of (asymmetric) key pairs.These are signing keys and verification keys for digital signatures and public keys and private keys for the public key encryption. We assume that the initial key exchange takes place, typically once, through a secure channel. In this way, information such as data encryption keys or migration notifications can be exchanged in a secure manner.

4 PROPOSED FRAMEWORK

In this work, we propose *PortableCloud* a framework that aims at providing cloud portability by seamlessly allowing data migration from one CSP to another one. Using *PortableCloud*, we not only address the vendor lock-in problem but also preserve the privacy of users by encrypting data before storing it in the cloud. *PortableCloud* enables users to share data with their contacts who may or may not belong to the same CSP.

Figure 2 depicts *PortableCloud*. In *PortableCloud*, the core system entities include a CSP, users, their contacts. A CSP is the core component of *PortableCloud* and the user data is stored at the CSP. The user data contains all information needed by the CSP to operate. For example, the user data contains the access policy and a command queue as a way to communicate with contacts. This encapsulated design of the user data makes it easy to migrate to a new CSP since the user data can be used as it is at the new CSP.

In this section, we provide an overview of the technical details of *PortableCloud*. We point out possible solutions and techniques to implement a portable cloud. After we discuss the CSP (Section 4.1), we describe how a new portable cloud account is set up (Section 4.2). In *PortableCloud*, users and contacts can communicate over a secure

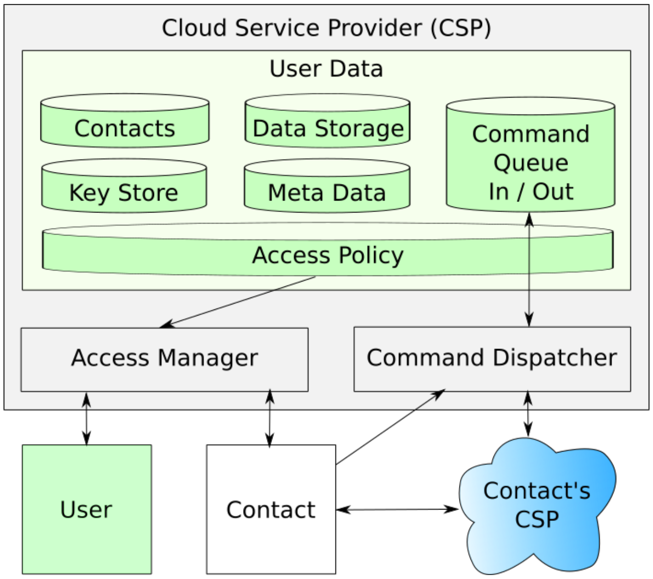

Fig. 2. *PortableCloud*: A user stores data at the CSP in an encrypted way. Only authorised contacts get access to the data at the CSP. The user and her contacts can exchange commands through their CSPs.

channel (Section 4.3). After a contact is established (Section 4.4), users and contacts can share and access the data (Section 4.5). *PortableCloud* also ensures data integrity and provenance (Section 4.6).

4.1 Cloud Service Provider (CSP)

A CSP is responsible for storing the user's data. It also regulates access to the data if a contact satisfies access policy specified by the user. Moreover, the CSP dispatches control commands between the user and her contacts, *e.g.,* commands to notify contacts about the migration. The user interacts with the CSP through a client such as a desktop, mobile app or web page. The CSP mainly consists of three main components including *User Data*, *Access Manager* and *Command Dispatcher*.

User Data. The user data is the core entity at the CSP that holds all the data of a single user. It also contains information the CSP needs to operate, *e.g.,* it includes the access control policy that is deployed to regulate access to the data. As we describe later, this simplifies the migration process since the user data is largely decoupled from the CSP. All sensitive elements of the user data are encrypted using a symmetric key encryption algorithm, where each element uses a separate symmetric key. The user data contains the following sub-components.

Data Storage. The data storage is a repository to store the data. Typically, it could be a database, a file system or a combination of both. Since the data in cleartext could compromise privacy of users, data elements (say files) are encrypted using different symmetric keys. This increases the security since contacts can only decrypt these entries for which they possess the corresponding decryption keys.

Meta Data. The meta data consists of structural information about the data, *e.g.,* directories and file names or table and column names of an encrypted database [Asghar et al., 2013b]. Furthermore, the meta data contains integrity and provenance information for the data stored in the data storage. Entries in the meta data are encrypted with encryption keys of the associated data.

Key Store. The key store is used to manage and store cryptographic keys at the CSP. It is important to note that all secret cryptographic keys are stored securely in an encrypted manner. Only the user can access the key store and get access to the keys using a password. This prevents the CSP accessing secret keys [Ferretti et al., 2014]. Thus, having access to the keys in the key store enables the user to decrypt all user data stored at the CSP.

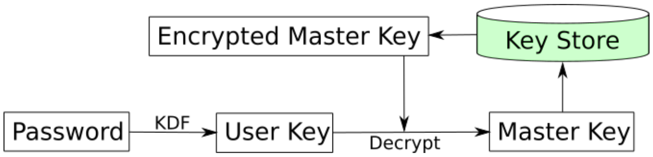

Fig. 3. A user key is generated from the user password using a Key Derivation Function (KDF). The user key is then used to decrypt the master key that allows the user access to the key store.

We consider that keys in the key store are encrypted with a symmetric master key. This master key, which could be realised as a role key, is stored encrypted as well. The master key can be decrypted by deriving a user key from the user's password using a Key Derivation Function (KDF) [Josefsson, 2011] (see Figure 3). This key chain approach is similar to the one used in the Linux Unified Key Setup (LUKS)[2] or adopted by Boxcryptor[3].

The key chain approach makes it easy to change a password for the key store. For changing the password, only the key store master key has to be re-encrypted with a user key derived from a new password. All other keys in the key store do not need to be re-encrypted.

Note that the security of the key store is limited by the complexity of the user password. The KDF can make it harder for brute-forcing attacks, *i.e.,* by using a salt and a high number of iterations. However, simple passwords can still be guessed easily. If a high degree of security is required, a password in form of a large cryptographic key should be used and stored safely at the user client.

Contacts. This entity contains information about all contacts known to the user. This information contains the CSP location of the contacts as well as contacts' public keys

[2] `https://gitlab.com/groups/cryptsetup`

[3] `https://www.boxcryptor.com`

and verification keys. Furthermore, each contact entry contains information to access and decrypt shared data that is located at the contact's CSP. All the contact information is stored encrypted.

Access Policy. The access control policy – or access policy in short – specifies what contacts are eligible to access at the user's CSP. In this work, we consider that access policies are readily available in the cleartext to allow the CSP to regulate access to the data. Without loss of generality, in case of sensitive access policies, we could employ encrypted policy enforcement mechanisms, such as [Asghar, 2013, Asghar et al., 2011, Asghar et al., 2013a, Asghar et al., 2015].

Command Queue. The command queue holds incoming or outgoing commands. Commands are generic messages that can be used to communicate requests between a user and her contacts. As discussed in Section 4.3, using commands, users can communicate securely with their contacts.

All commands in the incoming command queue are fetched and handled by the user. Outgoing commands are placed in the outgoing command queue together with a CSP address of the receiving contact. The CSP address is stored in cleartext in order to allow the CSP to dispatch the command.

Access Manager. The access manager is a sub-component of the CSP, which ensures that only authorised entities can get access to the data stored at the CSP. It authenticates users and contacts and provides access to the requested data, given the deployed access policies are satisfied.

Command Dispatcher. The command dispatcher is a sub-component of the CSP that dispatches commands from the outgoing command queue and aims at delivering them to the target CSP. If a command has been delivered successfully, it is removed from the queue. Moreover, this sub-component receives incoming commands and places them in the incoming command queue.

4.2 Account Creation

Once a new user signs up, a signing key pair as well as an encryption key pair are generated. Both key pairs are securely stored in the key store. As already explained in Section 4.1, a key chain is used, which starts from the user's password from which the user client derives the user key. Using this key chain, users can get access to the user data.

For authenticating the user to the CSP, we need to present user credentials. As a possible solution, we can use another password (different from one already mentioned above) but it would require the user to manage two passwords: one for authentication and another for decrypting the key chain, thus raising usability concerns. To avoid this usability issue, we propose authentication using the same password for deriving an additional authentication key from it. When signing up, the user chooses a different set of KDF parameters, *i.e.,* number of iterations and salt, to generate the authentication key.

4.3 Command Passing

A user needs a way to communicate with contacts at the same or different CSPs, say to establish a new contact (Section 4.4) or to share data with a contact (Section 4.5). In general, a direct peer-to-peer connection between the user and a contact is not always possible, in particular when the contact is offline. For this reason, the CSP is used to pass commands between communicating parties.

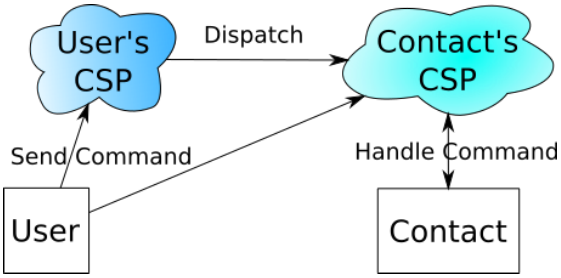

Fig. 4. The user can pass commands to the contact's CSP. These commands are handled by fetching them from the incoming command queue.

To send a command to a contact, the command is delivered to the contact's CSP as illustrated in Figure 4. The contact's CSP puts the command into the contact's incoming command queue. In case the contact's CSP is not available, the command is placed into the user's outgoing command queue and the user's CSP aims at delivering the command at a later time.

Commands are always signed by the sender and, whenever possible, encrypted with the public key of the receiver, assuming the public key has already been exchanged.

4.4 Contact Establishment

Data can be shared with contacts at the same or a different CSP. To establish a link between a user and an unknown contact, they have to exchange their verification keys and public keys. This key exchange could take place out-of-band, say using PGP [Garfinkel, 1995]. Alternatively, we can rely on Public Key Infrastructure (PKI), where a trust anchor, which is a root Certificate Authority (CA), issues X.509 certificates [Burr et al., 1996].

After a successful contact establishment, both the user and the new contact create a new contact entry in their contacts databases. Each contact entry contains the contact's keys as well as the CSP location of the contact.

4.5 Data Access and Sharing

Data Access. The user has full access to the user data by logging into the CSP using her user name and password (see Section 4.2). However, the access for contacts needs to be restricted and is managed using *access tokens*. The typical frameworks for access tokens include OAuth[4], OpenID Connect[5], where tokens are used to provide access to a service.

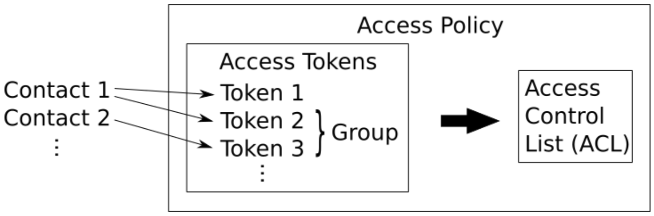

Fig. 5. Contacts can use tokens to gain access to the shared data. Tokens or groups of tokens are then associated with access rights defined in an ACL.

Typically, an access token is an identifier that is presented to a service provider to get access to requested services or resources. In the traditional access token model, there is no way to identify the requester. However, we consider special access tokens that are used to allow contacts access to specified resources. In the context of portable clouds, access tokens consist of two parts: a private signing key and a public verification key. The public part of the token, *i.e.,* the verification key, is stored in the user's access policy and is mapped to access rights specified in an Access Control List (ACL) [Sandhu and Samarati, 1994] (see Figure 5). The ACL can also be used to define group access, *i.e.,* multiple access tokens are mapped to the same access rights in the ACL. As described later, an eligible contact is in possession of the private part of the token, *i.e.,* the signing key.

A contact can access data by signing an access request using the private signing key. If the CSP can verify the access request using the public verification key in the ACL then the requested access is granted to the contact.

There are various scenarios in which access tokens are generated and how they are distributed. First, the user generates an access token when sharing data with a contact and sends the private signing key to the contact (see Section 4.3). Second, the contact generates an access token and sends the public verification key to the user. Third, one of the contact's signing keys is used in which case the user already knows the public verification key of the contact (see Section 4.4). In the following, we assume the first scenario, *i.e.,* the user generates an access token when sharing data.

Data Sharing. A user can share data with contacts located at the same or at a different CSP (see Figure 1). To access shared data, a contact needs an access token and an encryption key to decrypt the data (see Figure 6).

To share data with a contact, the user has to send the token as well as the encryption key to the contact. This is done using a secure command as described in Section 4.3. Moreover, the access rights in the access policy have to be updated for the used token.

[4] http://oauth.net

[5] http://openid.net/connect

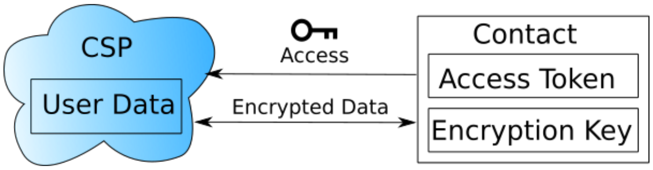

Fig. 6. A contact can access or modify data at a user's CSP. Therefore, the contact needs an access token and an encryption key.

If the contact declines the sharing offer, the changes made to the access policy are reverted. When revoking data access for a certain access token, the affected contacts are notified.

In case a contact wants to share data with the user and the user accepts the sharing offer, the user adds a new *shared data entry*. This shared data entry contains information about the shared data; the access token as well as the data encryption key. It is important to note that the contact's CSP location is already stored in the contacts database; thus, all information to access the contact's data is available.

4.6 Data Integrity and Provenance

An important property of a cloud storage is that users can ensure the integrity of their data stored at the CSP [Zhao et al., 2010], *i.e.,* detecting if potential attackers have tampered with the data at the CSP. Moreover, if the data is shared with a contact and the contact writes the data to the CSP, *i.e.,* modifies, adds or deletes the data, the user may want to ensure that the changes really originate from a certain contact [Asghar et al., 2012]. On the other hand, when the user writes the data, the user may want to certify that changes indeed originates from the user. This means not only data integrity is required but also data provenance is needed for a cloud storage.

To verify data integrity, *integrity information* is generated by the writer. One way to generate integrity information is to encrypt the data hash with the data encryption key and use this encrypted hash as integrity information [Hacigümüş et al., 2004]. The integrity information is stored in the meta data entity and can be accessed by users or contacts who can access the associated data. A user or a contact who is able to decrypt the data can also decrypt the integrity information and verify the data integrity by comparing the included hash to the hash of the actual data.

Integrity information does not help to ensure data provenance since multiple different writers may have write access to the same data. The writer also has to provide *provenance information*. The provenance information is stored along with the integrity information in the meta data. Same as the integrity information, the provenance information is encrypted with the data encryption key to prevent the CSP accessing the provenance data. The provenance information contains a hash that identifies the performed write operation, the ID of the writer and the ID of the hosting CSP user. Moreover, a time stamp can help to track when changes were made. Before encrypting the provenance information, the provenance information is signed with the writer's signing key.

The user can verify the data provenance by verifying that the hash of the write operation is compatible with the actual data. The receiver ID contained in the provenance information ensures that the write operation was indeed intended for the user. By verifying the writer's signature on the provenance information, the user can ensure that the changes were performed by the writer.

Unlike integrity information, the user may want to prevent contacts to access the provenance data. The provenance data contains information about with whom the user shares her data and the user may want to conceal this information [Asghar et al., 2012]. For example, if a user shares data with multiple contacts, the user may want to hide who else has access to the data. For that reason, the user can define in the access policy if contacts are allowed to access the provenance data from the meta data entity.

5 MIGRATION

The migration process of *PortableCloud* consists of two main steps. First, the user data has to be copied to the new CSP. Second, all contacts need to be notified about the new CSP. It is important to ensure that the migrations should be transparent for the contacts and there should be a minimal downtime.

Since the user data does not need to be adapted for the new CSP, the migration can take place through a direct data transfer between both CSPs. However, copying a large amount of user data can take a significant amount of time. For that reason, the data should be copied gradually. This can be done by first copying a snapshot of the user data and then successively copy new changes made during the migration. With the assumption that new changes are small, the time to synchronise the data with the new CSP during the migration is small.

One problem that can affect the migration is ongoing write or read transactions that are performed by contacts. A conservative approach is to block new changes at the old CSP and wait unless all the data is migrated. Alternatively, we can imagine more sophisticated approaches that are able to handle ongoing changes during and after the migration.

Once all data is copied to the new CSP, the contacts need to be notified about the migration. This is done by sending them *migration commands* that contain the location of the new CSP. A problem that can occur here is that a contact might not be reached. One reason for that could be the temporary unavailability of the contact's CSP. However, since the migration command is in the outgoing command queue, the new CSP will try to deliver the message at a later point. Another situation when a migration command cannot be delivered is when the user and a contact both migrate to a new CSP at the same time. In this case, there is no easy way to determine the CSP location of the migrating contact. For this reason, the old CSP can be configured to point to the new CSP location when contacts try to communicate with the old CSP. One possible approach is to introduce one or more central name servers where users can register their CSP location.

If a user receives a migration command from a migrated contact, the new contact's CSP location has to be updated in the user's contacts list. Furthermore, the user has to verify that undelivered outgoing commands to the migrated contact are updated to target the contact's new CSP.

5.1 Migration Costs

For enterprises as well as individuals, the costs and services of a CSP are important. If a preferable CSP (say based on various factors such as quality of service or costs) is available, the user may consider migrating to this CSP.

Data Sharing Systems. In the following, we discuss the migration between two different data sharing systems. It also includes the migration to the portable cloud architecture. We assume that the user encrypts data on the client side to prevent the CSP accessing the data.

One of the major costs includes set up costs, such as initial setup fees for the new CSP. There are various sources of costs when transferring data from the old to the new CSP. First, one or both of the involved CSPs may have data transfer fees. Second, it might not be possible to transfer the data directly between both CSPs and the user needs a local data storage to copy the data. For instance, data formats or databases may be incompatible, data requires re-encryption or there is a lack of APIs to transfer data directly.

Once the data is transferred to the new CSP, connections to old contacts have to be re-established and access policies have to be set up. In general, there is no automatic way to convert the old access policy to a new system. For this reason, the access policy has to be verified manually, which can be an expensive and erroneous process, *e.g.,* due to human errors crucial data could accidentally be leaked to wrong contacts.

Portable Clouds. For the migration of the portable clouds, there may be set up and data transfer costs. Even the small migration downtime for the portable clouds could lead to further costs.

PortableCloud minimises the cost described above. Since the data can be transferred directly between the old and the new CSP, expensive data re-encryption and round trips to the user's local storage could be eliminated. Furthermore, *PortableCloud* ensures that contacts can still access the shared data and no new encryption keys have to be exchanged. This not only minimises the service downtime but also is fail-safe against human errors, *i.e.,* the old access policies are re-used at the new CSP.

The user as well as all contacts do not need to update or reconfigure their client software since the migration process is transparent. This eliminates support costs and expensive software adoptions.

5.2 Migration Agent

There are various decision making and other tools that could assist during the migration process [Satzger et al., 2013,Ward et al., 2010,Khajeh-Hosseini et al., 2011]. Like these tools, we use a migration agent in *PortableCloud.* The migration agent calculates costs based on various parameters of interest, which includes, but are not limited to, historical growth pattern, manual input or a combination of both. The migration agent assists users in providing statistics about data usage, forecasting and listing alternative CSPs that can offer similar or even better services. If the agent finds a better CSP, it suggests it to the user as a migration option. For that, the cloud agent maintains a knowledge base of alternative CPSs in real-time. This knowledge base is updated regularly by services that host the migration agent.

A core aspect when considering migration of the portable clouds is cost. The costs identified in Section 5.1, *e.g.,* initial set up and data transfer costs, are taken into account. Another interesting parameter is the migration time. The migration agent can estimate how long a migration will take, *e.g.,* how much time the account set up and the data transfer will take. This helps the user in estimating when the new service of the CSP is available.

The migration agent also estimates usage patterns and notifies a user about possible performance problems and issues. These problems could be a result of lacking or surplus of data storage, transfer problems with the users/contacts, or stability and reliability issues with the CSP. For example, if for a certain period of time the user consumes less storage space than she pays for, the migration agent analyses if there are more suitable (*i.e.,* economical) options available.

The migration agent also considers different factors such as customer satisfaction, reputation or legal issues with the CSP. However, these factors are subjective and have to be considered carefully. Furthermore, the migration agent helps users in finding better service plans at the current CSP, if available.

6 DISCUSSION

One goal of *PortableCloud* is to maintain privacy of users. In this section, we discuss what information the CSP can gain about the user and what information is concealed from the CSP. Moreover, we discuss requirements and solutions for an enterprise that uses the portable clouds.

Privacy. There is some general knowledge a CSP has about its users. For instance, when registering with a CSP, information such as the user name / login name, email address, phone numbers, postal address or payment details may be revealed to the CSP.

All data the user stores at the CSP is encrypted and can only be read by the user who has the corresponding key. In *PortableCloud*, the meta data is also protected. Thus, the CSP cannot learn any sensitive user data.

Outgoing commands contain the target CSP in order to deliver a command to a certain contact. This may reveal the identity of contacts. Although all information about contacts is stored encrypted, the CSP can derive information about the number of contacts of a user. To address this issue, Oblivious RAM (ORAM) [Stefanov et al., 2013, Goldreich and Ostrovsky, 1996] or related techniques may be necessary.

The access policy maps access tokens to an access control map, which may reveal information to the CSP. For example, the CSP can analyse how many access tokens exist for a certain data entry and may derive information about the number of contacts or the importance of the data entry.

Note that the CSP can analyse traffic from/to the user data, which could also reveal information about the stored data as well as about the contacts [Gong et al., 2010, Raymond, 2001].

Enterprises. In *PortableCloud*, as described above, users control their data and manage contacts they share their data with. However, for commercial enterprises, this model might not be an ideal option. In the following, we describe what requirements an enterprise may have concerning portable clouds and how *PortableCloud* can be customised to fulfil these tailored requirements.

An enterprise usually has a number of employees and there are certain restrictions on how data can be shared with internal and external contacts. For this reason, the enterprise needs a way to manage their employees. To do so, the enterprise takes the role of an admin user who can manage a group of users at the CSP (see Figure 7). The admin user has several privileges such as:

- Administration of new users, *i.e.,* creation and deletion.
- Control data access among users of the enterprise and external contacts.
- Prevent users to migrate their user data to another CSP.

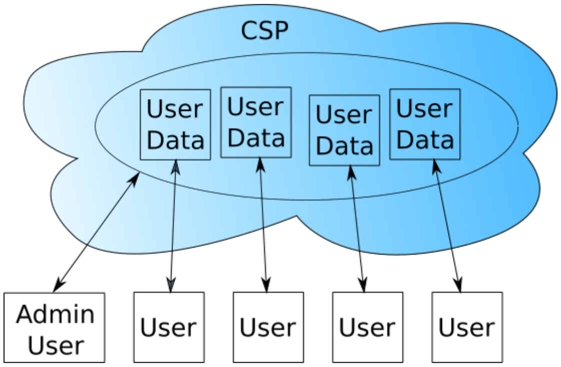

Fig. 7. An enterprise can administrate and manage multiple users (*e.g.,* its employees). The enterprise may have special access rights to its employees' data and key stores.

In general, an enterprise can require access to all data produced by their employees. The enterprise can require employees to enable their admin access to their secret key. This would also allow the admin to reset their secret keys. Since employees only use their own personal password to encrypt their secret key (see Section 4.1), the personal password is not revealed to the employer. This is important in case the employee uses this password also for other purposes.

Another approach of using *PortableCloud* in an enterprise is to allow multiple clients to access a single portable cloud account. This is easily possible since the key store supports multiple access passwords. However, since all employees have the same access rights, all data can be accessed by all employees. Thus, this solution is only suitable for small enterprises.

Data Sharing with External Contacts. *PortableCloud* only allows sharing data with contacts that have a *PortableCloud* account at the same or a different CSP. It is often desirable to share data with external contacts, those who do not have a *PortableCloud* account. A common solution for this is public links. By sharing public links with external contacts, *e.g.,* via email, external contacts can gain access to shared data. For example, in Dropbox[6], files can be shared with external contacts using a public hyperlink. Overleaf[7] even allows external contacts to edit documents shared through a public hyperlink.

In general, a public link contains an access token that ensures that only contacts who know the link are able to access the shared data. By choosing a sufficiently large random access token, it becomes very difficult for an adversary to gain access to the shared data. Thus, public links provide fairly good access control.

A disadvantage of public links is that, they are not easy to remember due to the embedded access token. External contacts have to manage their public links manually. Another problem with public links occurs if the shared data is encrypted. In this case, also the decryption key has to be shared with external contacts. For example, to access encrypted data through a web application, the external contact would need to provide the decryption key to the web application. Note that, in the case of a web application, the entity who provides the web application has to be trusted for not leaking the decryption key from the web application to an adversary. Another solution is to drop the privacy requirement for the data that is shared through public links and reveal the decryption key to the CSP. However, this is usually not desirable.

Data Update Notifications. When having access to the shared data, it becomes interesting to know when the data has been updated at the CSP. This is because polling for data updates can be expensive especially when monitoring a huge amount of data. For example, if a user has access to multiple files from different contacts, frequently checking for data updates becomes expensive for the user as well as for the involved CSPs. A more efficient way to monitor data updates is a publish-subscribe model [Cabrera et al., 2001, Cooper et al., 2008]. Here, a subscriber can register with a publisher and the publisher notifies the subscriber if updates are available.

The simplest approach for a user to notify contacts about data updates is to send a *data update command* to all contacts who have access to the updated data. This approach does not leak information about who has access to the data. However, contacts do not get notified immediately when other contacts modify data because the contact who changed the data may not know who else has access to the data and thus is not able to send data update commands to other sharing contacts.

A different approach is to let the server notify contacts when data has been updated. For example, a contact registers with the user's CSP to indicate that she is interesting in updates on certain data sets. When data has been changed the CSP automatically sends data update commands to the registered contacts. A problem with this approach is that the CSP can gain information on who has access to which data. Protection of this information requires some other privacy-preserving techniques [Pal et al., 2012].

[6] `dropbox.com`

[7] `www.overleaf.com`

7 RELATED WORK

Although migrating a system to the cloud is a challenging task, migration also brings scalability while offering flexible pricing options [Zhao and Zhou, 2014]. Migrating a local service to the cloud can reduce the cost to run and maintain servers but can also increase the dependency on external third parties and a potential deterioration of the service quality due to less control over the system [Khajeh-Hosseini et al., 2011].

For enterprises, it is not easy to decide if the migration from their IT system to the cloud is really beneficial. Cloud Genius assists users in finding an optimal CSP that provides IaaS, *i.e.,* it finds the IaaS that is able to run a certain VM image at better service conditions [Menzel and Ranjan, 2012]. The problem of vendor lock-in can be addressed by using unified programming APIs and domain-specific languages to model application components and cloud requirements [Satzger et al., 2013]. In a so-called meta-cloud, an agent continuously checks for alternative CSPs with better conditions for the specified requirements [Satzger et al., 2013].

One way to share data is to use a distributed peer-to-peer data sharing system, such as PeerDB [Ng et al., 2003]. However, the data in PeerDB is not encrypted. Moreover, for sharing data, both peers are expected to be online.

Various security and privacy issues in cloud computing have been identified [Takabi et al., 2010]. When transferring the data from/to the cloud, confidentiality and integrity must be ensured. When sharing data with other parties, there must be mechanisms to control access rights. To ensure privacy when storing data in the cloud, the usual way is to encrypt data. However, users may not have enough expertise to manage their keys. The data integrity can usually not be verified on the cloud storage without transferring the data to a local machine. When deleting data in the cloud, the user usually cannot ensure that no data copies remain at the CSP. One way of dealing with privacy issues is to keep users anonymous while storing the user's data in cleartext in the cloud [Khan and Hamlen, 2012]. K2C allows users to share encrypted data with other users but users have to manage their encryption keys in a local key store [Zarandioon et al., 2012]. A more convenient approach is to store the encryption keys in an encrypted key store in the cloud [Ferretti et al., 2014]. Even when data is encrypted, it is possible to perform a search query on the encrypted data while respecting multi-user access policies [Asghar et al., 2013b].

The cloud storage system DepSky [Bessani et al., 2011] stores encrypted and signed data at multiple CSPs. DepSky uses a secret sharing scheme [Butoi and Tomai, 2014], which means that shares of the secret key are distributed to different CSPs. While DepSky allows users to replicate data at different CSPs, it does not offer any contact management.

There are various popular cloud sharing systems available. The cloud software ownCloud[8] allows users to setup a personal cloud server. However, while ownCloud enables public data access, private data can only be shared securely with users of the same server and not with users of other ownCloud servers. ownCloud only supports server side encryption, which requires trusting the server that hosts the ownCloud instance.

[8] https://owncloud.com/

Data sharing platforms, such Boxcryptor[9], support the client side encryption. However, these services neither support migration to another CSP nor do they allow private data sharing with users of other CSPs.

Mona allows users to share data with contacts and revoke access if necessary [Liu et al., 2013]. While the identity of a contact is concealed from the CSP, the user knows about the provenance of the data. To define an access policy, a simple Role-Based Access Control (RBAC) mechanism can be used. Here, roles can be granted and revoked if necessary [Sandhu et al., 1996]. Moreover, hierarchical attribute-based encryption can be used to control and revoke data access [Wang et al., 2011].

When establishing a contact, the public keys of both parties have to be exchanged. This key exchange is vulnerable to man-in-the-middle attacks. SafeSlinger enables an easy and secure exchange of public keys between contacts as long as there a secure channel between them, *i.e.,* they can exchange a simple word phrase in person or via other channels [Farb et al., 2013].

8 CONCLUSIONS AND FUTURE WORK

In this chapter, we addressed the problem of vendor lock-in, which makes it difficult for cloud users to migrate to an alternative CSP because the data cannot easily be transferred to a new CSP and data shared with contacts at the old CSP may become inaccessible after the migration. To fill the gap, we presented *PortableCloud*, a framework that makes it possible to migrate a data sharing system to a new CSP. In *PortableCloud*, users can share data with contacts located at the same or at different CSPs. *PortableCloud* provides mechanisms to store the data in an encrypted manner.

We discussed the cost of migrating a portable cloud and various aspects, necessary for designing *PortableCloud*. We described a migration agent that assists users in automatically finding a suitable CSP that could satisfy their needs.

As future work, we plan to complete the implementation of *PortableCloud*. Furthermore, investigating accountability aspects of portable clouds would be an interesting research direction.

References

Armbrust et al., 2010. Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2010). A view of cloud computing. *Commun. ACM*, 53(4):50–58.

Asghar, 2013. Asghar, M. R. (2013). *Privacy Preserving Enforcement of Sensitive Policies in Outsourced and Distributed Environments*. PhD thesis, University of Trento.

Asghar et al., 2011. Asghar, M. R., Ion, M., Russello, G., and Crispo, B. (2011). ESPOON: Enforcing encrypted security policies in outsourced environments. In *The Sixth International Conference on Availability, Reliability and Security*, ARES'11, pages 99–108.

Asghar et al., 2012. Asghar, M. R., Ion, M., Russello, G., and Crispo, B. (2012). Securing data provenance in the cloud. In *Open Problems in Network Security*, volume 7039 of *Lecture Notes in CS*, pages 145–160.

[9] `https://www.boxcryptor.com`

Asghar et al., 2013a. Asghar, M. R., Ion, M., Russello, G., and Crispo, B. (2013a). ESPOON$_{ERBAC}$: Enforcing security policies in outsourced environments. *Elsevier Computers & Security (COSE)*, 35:2–24.

Asghar et al., 2015. Asghar, M. R., Russello, G., and Crispo, B. (2015). E-GRANT: Enforcing encrypted dynamic security constraints in the cloud. In *Future Internet of Things and Cloud (FiCloud)*, pages 135–144. Special Track on Security, Privacy and Trust.

Asghar et al., 2013b. Asghar, M. R., Russello, G., Crispo, B., and Ion, M. (2013b). Supporting complex queries and access policies for multi-user encrypted databases. CCSW '13, pages 77–88.

Bessani et al., 2011. Bessani, A., Correia, M., Quaresma, B., André, F., and Sousa, P. (2011). Depsky: Dependable and secure storage in a cloud-of-clouds. EuroSys '11, pages 31–46.

Burr et al., 1996. Burr, W. E., Nazario, N. A., and Polk, W. T. (1996). A proposed federal PKI using X.509 v3 certificates. *NIST*.

Butoi and Tomai, 2014. Butoi, A. and Tomai, N. (2014). Secret sharing scheme for data confidentiality preserving in a public-private hybrid cloud storage approach. UCC'14, pages 992–997.

Cabrera et al., 2001. Cabrera, L. F., Jones, M. B., and Theimer, M. (2001). Herald: Achieving a global event notification service. In *Hot Topics in Operating Systems, 2001. Proceedings of the Eighth Workshop on*, pages 87–92. IEEE.

Cooper et al., 2008. Cooper, B. F., Ramakrishnan, R., Srivastava, U., Silberstein, A., Bohannon, P., Jacobsen, H.-A., Puz, N., Weaver, D., and Yerneni, R. (2008). Pnuts: Yahoo!'s hosted data serving platform. *Proc. VLDB Endow.*, 1(2):1277–1288.

De Capitani di Vimercati et al., 2008. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Pelosi, G., and Samarati, P. (2008). Preserving confidentiality of security policies in data outsourcing. WPES '08, pages 75–84.

De Chaves et al., 2011. De Chaves, S., Uriarte, R., and Westphall, C. (2011). Toward an architecture for monitoring private clouds. *Communications Magazine, IEEE*, 49(12):130–137.

Farb et al., 2013. Farb, M., Lin, Y.-H., Kim, T. H.-J., McCune, J., and Perrig, A. (2013). Safeslinger: Easy-to-use and secure public-key exchange. MobiCom '13, pages 417–428.

Ferretti et al., 2014. Ferretti, L., Colajanni, M., and Marchetti, M. (2014). Distributed, concurrent, and independent access to encrypted cloud databases. *Parallel and Distributed Systems*, 25(2):437–446.

Garfinkel, 1995. Garfinkel, S. (1995). *PGP: pretty good privacy*.

Goldreich and Ostrovsky, 1996. Goldreich, O. and Ostrovsky, R. (1996). Software protection and simulation on oblivious RAMs. *J. ACM*, 43(3):431–473.

Gong et al., 2010. Gong, X., Kiyavash, N., and Borisov, N. (2010). Fingerprinting websites using remote traffic analysis. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 684–686.

Hacigümüş et al., 2004. Hacigümüş, H., Iyer, B., and Mehrotra, S. (2004). Ensuring the integrity of encrypted databases in the database-as-a-service model. In *Data and Applications Security 17*, volume 142, pages 61–74.

Joint et al., 2009. Joint, A., Baker, E., and Eccles, E. (2009). Hey, you, get off of that cloud? *Computer Law & Security Review*, 25(3):270 – 274.

Josefsson, 2011. Josefsson, S. (2011). PKCS# 5: Password-Based Key Derivation Function 2 (PBKDF2) test vectors. Technical report.

Khajeh-Hosseini et al., 2011. Khajeh-Hosseini, A., Sommerville, I., Bogaerts, J., and Teregowda, P. (2011). Decision support tools for cloud migration in the enterprise. In *Cloud Computing (CLOUD)*, pages 541–548.

Khan and Hamlen, 2012. Khan, S. and Hamlen, K. (2012). Anonymouscloud: A data ownership privacy provider framework in cloud computing. In *Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 170–176.

Liu et al., 2013. Liu, X., Zhang, Y., Wang, B., and Yan, J. (2013). Mona: Secure multi-owner data sharing for dynamic groups in the cloud. *Parallel and Distributed Systems*, 24(6):1182–1191.

Menzel and Ranjan, 2012. Menzel, M. and Ranjan, R. (2012). CloudGenius: Decision support for web server cloud migration. WWW '12, pages 979–988.

Ng et al., 2003. Ng, W. S., Ooi, B. C., Tan, K.-L., and Zhou, A. (2003). PeerDB: A P2P-based system for distributed data sharing. In *Data Engineering*, pages 633–644.

Pal et al., 2012. Pal, P., Lauer, G., Khoury, J., Hoff, N., and Loyall, J. (2012). P3s: A privacy preserving publish-subscribe middleware. In *Middleware 2012*, pages 476–495.

Raymond, 2001. Raymond, J.-F. (2001). Traffic analysis: Protocols, attacks, design issues, and open problems. In *Designing Privacy Enhancing Technologies*, pages 10–29. Springer.

Sandhu and Samarati, 1994. Sandhu, R. and Samarati, P. (1994). Access control: principle and practice. *Communications Magazine, IEEE*, 32(9):40–48.

Sandhu et al., 1996. Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. (1996). Role-based access control models. *Computer*, 29(2):38–47.

Satzger et al., 2013. Satzger, B., Hummer, W., Inzinger, C., Leitner, P., and Dustdar, S. (2013). Winds of change: From vendor lock-in to the meta cloud. *IEEE Internet Computing*, 17(1):69–73.

Stefanov et al., 2013. Stefanov, E., van Dijk, M., Shi, E., Fletcher, C., Ren, L., Yu, X., and Devadas, S. (2013). Path ORAM: An extremely simple oblivious ram protocol. CCS '13, pages 299–310.

Takabi et al., 2010. Takabi, H., Joshi, J. B., and Ahn, G.-J. (2010). Security and privacy challenges in cloud computing environments. *Security & Privacy*, 8(6):24–31.

Wang et al., 2011. Wang, G., Liu, Q., Wu, J., and Guo, M. (2011). Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers. *Computers & Security*, 30(5):320 – 331.

Ward et al., 2010. Ward, C., Aravamudan, N., Bhattacharya, K., Cheng, K., Filepp, R., Kearney, R., Peterson, B., Shwartz, L., and Young, C. (2010). Workload migration into clouds challenges, experiences, opportunities. In *Cloud Computing (CLOUD)*, pages 164–171.

Zarandioon et al., 2012. Zarandioon, S., Yao, D., and Ganapathy, V. (2012). K2C: Cryptographic cloud storage with lazy revocation and anonymous access. In *Security and Privacy in Communication Networks*, volume 96, pages 59–76.

Zhao et al., 2010. Zhao, G., Rong, C., Li, J., Zhang, F., and Tang, Y. (2010). Trusted data sharing over untrusted cloud storage providers. In *Cloud Computing Technology and Science (CloudCom)*, pages 97–103.

Zhao and Zhou, 2014. Zhao, J.-F. and Zhou, J.-T. (2014). Strategies and methods for cloud migration. *International Journal of Automation and Computing*, 11(2):143–152.