Privacy-preserving Data Sharing in Portable Clouds

Clemens Zeidler and Muhammad Rizwan Asghar
University of Auckland, 38 Princes Street, Auckland 1142, New Zealand

Keywords: Portable Cloud, Privacy, Data Sharing, Data Migration, Migration Costs, Migration Agent.

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 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.

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. 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.

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 between 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 paper, 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 condition 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 paper, we propose *PortableCloud*, an architecture that targets 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 3). To preserve 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.

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

Zeidler, C. and Asghar, M.
Privacy-preserving Data Sharing in Portable Clouds.
In *Proceedings of the 6th International Conference on Cloud Computing and Services Science (CLOSER 2016) - Volume 2*, pages 274-281
ISBN: 978-989-758-182-3

Our contributions can be summarised as follows:

- A proposal of a novel privacy-preserving portable cloud architecture *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.

2 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 to store the data and run 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 policies at 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. The user 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 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 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 key pairs for asymmetric cryptography. These are signing keys and verification keys for digital signatures and public keys and private keys for public key encryption.

3 PROPOSED ARCHITECTURE

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.

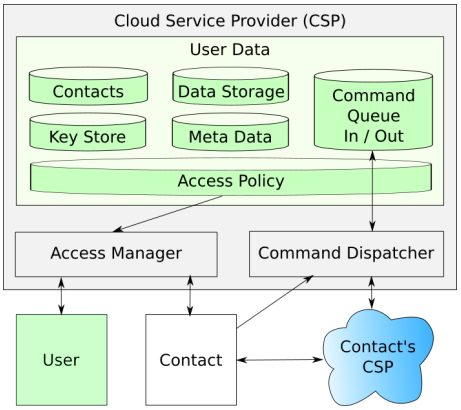

Figure 1: *PortableCloud*: A user stores data at the CSP in an encrypted way. Only authorised contacts get access to the data at the CSP. The user and her contacts can exchange commands through their CSPs.

Figure 1 depicts *PortableCloud*. In *PortableCloud*, the core system entities include a CSP, users, their contacts (and the CSPs of contacts).

3.1 Cloud Service Provider (CSP)

A CSP is responsible to store the data. It also regulates access to the data if a contact satisfies access policy specified by the user. The user interacts with the CSP through a client such as a desktop, a mobile app or a 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 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 the symmetric key encryption algorithm while 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 sym-

metric 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., 2013). 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.

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). This key chain approach is similar to the one used in the Linux Unified Key Setup (LUKS)[2] or for Boxcryptor[3].

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 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 policy specifies what contacts are eligible to access data 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).

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 3.3, using commands, users can communicate securely with their contacts.

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

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

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.

3.2 Account Creation

Once a new user signs up, a signature key pair and an encryption key pair are generated. Both key pairs are securely stored in the key store. As already explained in Section 3.1, 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 concern. To avoid this usability issue, we propose authenticating using the same password but 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.

3.3 Command Passing

A user needs a way to communicate with contacts at the same or different CSPs, say to establish a new contact (Section 3.4) or to share data with a contact (Section 3.5). In general, a direct peer-to-peer connection between the user and a contact is not possible, *i.e.,* when the contact is offline. For this reason, the CSP is used to pass commands between communicating parties.

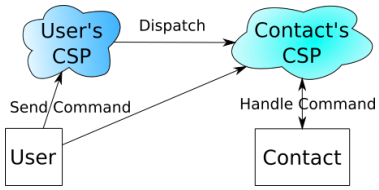

Figure 2: The user can pass commands to the contact's CSP. These commands are handled by fetching them from the incoming command queue.

To send a command to a contact, the command is delivered to the contact's CSP as illustrated in Figure 2. 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, *i.e.,* when the public key has been exchanged.

3.4 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 containing the contact's keys and CSP location. The contact entry contains the contact's keys as well as the CSP location of the contact.

3.5 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 3.2). 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.

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). 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.

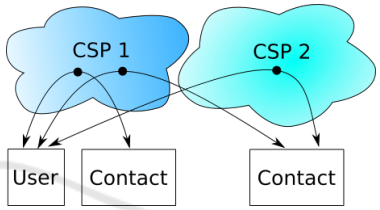

Figure 3: Data sharing between a user and her contacts who may or may not be at the same CSP.

Data Sharing. A user can share data with contacts located at the same or at a different CSP (see Figure 3). To access shared data, a contact needs an access token and an encryption key to decrypt the data (see Figure 4).

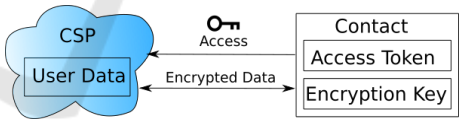

Figure 4: A contact can access or modify data at a user's CSP. Therefore, the contact needs an access token and an encryption key.

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 3.3. Moreover, the access rights in the access policy have to be updated for the used token.

3.6 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 data is shared with a contact and the contact writes data to the CSP, *i.e.,* modifies, adds or deletes

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

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 data, the user may want to certify that changes indeed originates from the user. This means not only data integrity but also data provenance is an important property for a cloud storage.

To be able to verify data integrity and provenance a user or contact has to provide integrity and provenance information (Hacigümüş et al., 2004). This information is stored in the meta data entity and can be accessed by users or contacts who can access the associated data.

4 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 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 cannot be reached. One reason for that could be 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.

4.1 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 set up 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 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 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 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.

4.2 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 service. 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 4.1, *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 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 to find better service plans at the current CSP, if available.

5 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 at a CSP, information such as the user name and 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.

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 5). The admin user has several privileges, such as the administration of new users, *i.e.,* creation and deletion and controlling data access between users/contacts.

In general, an enterprise can require access to all data produced by their employees. The enterprise can require employees to enable their admin access

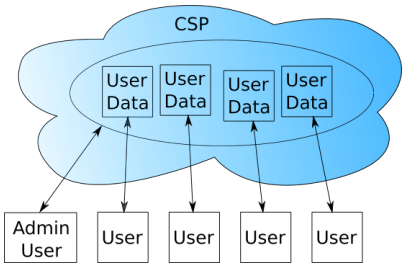

Figure 5: An enterprise can administrate and manage multiple users (*e.g.,* its employees). The enterprise may have special access rights to its employees' data and key stores.

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 3.1), the personal password is not revealed to the employer. This is important in case the employee uses this password also for other purposes.

6 RELATED WORK

The difficult task of migrating a system to the cloud has the advantage of being scalable while having a low maintenance and flexible pricing options (Zhao and Zhou, 2014). Migrating a local service to the cloud can reduce cost to run and maintain servers but can also increased the dependency on external third parties and a deterioration of the service quality because of less control over the system (Khajeh-Hosseini et al., 2011).

For enterprises, it is not easy to decide if a migration from their IT system to the cloud is 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, 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). 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., 2013).

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 on 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[6] allows users to setup a personal cloud server. Data sharing platforms, such Boxcryptor[7], 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 contacts 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).

7 CONCLUSIONS AND FUTURE WORK

In this paper, we addressed the problem of vendor lock-in, which makes it difficult for cloud users to migrate to an alternative CSP because 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*, an architecture 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 data in an encrypted manner.

We discussed the cost of migrating a portable

[6]https://owncloud.com/
[7]https://www.boxcryptor.com

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*. We believe that an analysis of the migration performance will confirm the feasibility of proposed portable clouds. Furthermore, investigating accountability aspects of portable clouds would be an interesting research direction.

REFERENCES

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

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.

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

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

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.

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

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

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

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

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

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

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.

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

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