TwinCloud: Secure Cloud Sharing Without Explicit Key Management

Kemal Bicakci*, Davut Deniz Yavuz†, Sezin Gurkan‡
Department of Computer Engineering
TOBB University of Economics and Technology
*bicakci@etu.edu.tr, †st111101036@etu.edu.tr, ‡st111101054@etu.edu.tr

***Abstract*—In this paper, we propose TwinCloud as a client-side solution providing a secure system to users without compromising the usability of cloud sharing. TwinCloud brings a novel solution to the complex key exchange problem and provides a simple and practical approach to store and share files by hiding all the cryptographic and key-distribution operations from users. Serving as a gateway, TwinCloud stores the encryption keys and encrypted files in separate clouds which ease the secure sharing without a need for trust to either of the cloud service providers with the assumption that they do not collude with each other. TwinCloud is a lightweight application and available as open-source.**

***Keywords*-cloud security; cloud storage; key management**

I. Introduction

Cloud storage is a storage model in which the data is stored in multiple remote servers and locations owned by a hosting company like Dropbox or Google. These cloud providers must keep the data accessible as well as confidential from other users. They offer services like easy file storage, sharing, syncing and collaboration among cloud users. Because of these useful services, today, both regular consumers and business organizations are widely using the cloud storage. However, while cloud storage services provide these services, more and more personal and confidential data is being exposed to privacy and security risks.

Cloud providers have developed new security methods to protect users data from attackers. For example, Dropbox and iCloud started to use two-step verification where the idea is to combine something you know (password) with something you have (phone) to add an extra layer of security [1], and also Dropbox uses Secure Sockets Layer (SSL)/Transport Layer Security (TLS) to protect data in transit between client apps and Dropbox servers [2]. Moreover, Google Drive stores the data randomly distributed across multiple machines which provide another layer of security [3].

However, these security mechanisms are not sufficient and only protect users from third party unauthorized access, not from the cloud storage providers access to confidential files. Google states in its Terms of Service that when you upload or submit a file, you give Google a license to use, modify, publish and distribute the content of the file [4]. Furthermore, Google analyzes the file content and uses for advertising. Dropbox has also a similar term of service for the files that you put in their storage. Moreover, according to Snowden's documents [5], NSA has a secret program called Prism which the agency collects sensitive data from Google, Facebook, Apple, Yahoo and other US Internet giants.

In order to protect the customers data stored in the cloud, two models are used among cloud service providers and third party cloud applications:

Server-side encryption protects the data from attackers, but cloud providers have the encryption key and the files get decrypted on their servers every time they are accessed. Moreover, their administrators can see users files, and so can anyone who manages to gain access to their systems. For instance, in 2010, a Google employee accessed several Gmail and Hangouts accounts to spy and harass people [6]. Thus, server-side encryption is not sufficient itself for security and need to be supported by other security measures like client side encryption.

Client-side encryption eliminates the above problem since files are encrypted before uploading to remote cloud servers with an encryption key that only a specific user knows. There are several commercial cloud encryption programs such as nCrypted, Sookasa, Tresorit and BoxCryptor which provide client-side encryption. However, this method reduces the usability of cloud services. With client-side encryption, users could not share their files as easily as before because they also need to share the encryption key with other users from a secure channel. In order to solve the key distribution problem, several PKI-based methods were proposed [7], [8], [9], [10]. However, PKI-based solutions have some drawbacks. They are costly because they need to get a certificate from a CA. On the overall, it is well-documented that managing cryptographic keys is not manageable for average computer users [11].

To share files, either encrypted or not, there are several methods used by cloud providers. File owners can use one of the following sharing methods to share their files with business partners, friends and coworkers [12]:

1) Public sharing: File is shared with a public URL that everyone can access.
2) Secret-URL sharing: File is shared by sending a private sharing URL to specific users.
3) Private sharing: File owner must specify who can access the shared file and cloud service provider authenticates the specified users while accessing the file.

Both Dropbox and Google Drive supports all of these sharing methods (however as of this writing private and public sharing is only applicable for folders, not for files on Dropbox).

The sharing methods listed above are used only for sharing the original file and are not yet used for key sharing. Thus, after encryption, using cloud providers services, it is only possible to share the encrypted file. Encryption key needs to be shared from another channel. Finding a secure channel to share the key is a problem with serious usability challenges.

In this paper, we present a novel solution, TwinCloud, that uses client-side encryption and private sharing method. Cloud users could continue to use the cloud services as before in a simple way. Users are able to share their files securely using the TwinCloud application which maintains the encrypted file and the encryption key in separate clouds where users can do all file operations from this single application without a need for explicit key management.

TwinCloud generates a key and encrypts the file with this key before storing in cloud servers. Then, it uploads the encryption key to a cloud and encrypted file to another cloud. Users easily share a file to a specific user by using the TwinCloud's user interface while TwinCloud, in the back-end, shares the encryption key and the encrypted file to the specific user using cloud providers' file sharing features.

The rest of this paper is organized as follows: Section II discusses the related work. Section III describes our solution and the properties of TwinCloud. Section IV provides a comparison of our solution to other solutions and discusses several extensions and provides ideas for future work. And finally, Section V concludes the paper.

II. Related Work

This section examines preceding industrial and academic work related to secure file sharing on clouds.

Boxcryptor [9] is a widely used commercial tool to protect files in the cloud and it supports many cloud service providers such as Dropbox, Google Drive, Microsoft OneDrive and Box. It uses AES with a key length of 256 bits, CBC (Cipher Block Chaining), PKCS7 padding and RSA with a key length of 4096 bits and OAEP padding. Boxcryptor uses a trusted CA and the following steps to share file between Alice and Bob [13]:

1) Alice requests Bob's public key from the Boxcryptor Key Server.
2) Alice encrypts the file key with Bob's public key.
3) Alice writes the new encrypted key file.
4) The cloud storage provider syncs the modified encrypted key file.
5) Bob uses his private key to decrypt the file key.
6) Bob uses the file key to decrypt the file.

Tresorit [10] is a cloud provider that uses client-side encryption. They use PKI and PKC to share the encryption key. Tresorit does not need a trust to any other certificate issuer, certificates are issued by Tresorit company itself as the Tresorit User CA certificate. Users can share files by creating Tresor (a shareable secure online folder) that contains the files they want to share or by creating and sending encrypted links to other users.

There are sharing methods proposed in the academic literature which have used client-side encryption and nearly all of these methods [7], [14], [15] use PKC or PKI-based solutions. However, there are some disadvantages of these solutions in terms of usability and security. Managing a trusted PKI is costly and hard to maintain. It also burdens the users with handling cryptographic operations. Moreover, if an attacker manages to obtain the private key of a CA certificate, the attacker can see all the files and its contents. In above cases, the attacker could potentially be someone from the cloud service providers. When the certificates were issued by cloud providers, this means that they also can view user data whenever the keys are used for encryption/decryption purposes [16].

By realizing this, we design TwinCloud that does not require a trusted third party like a CA and does not use a PKI structure or PKC. It uses only symmetric key cryptography. It also does not require users to handle key management.

III. Solution

A. *Solution in a Nutshell*

TwinCloud is a lightweight application that uses client-side encryption and private sharing.

Our solution requires at least two cloud service providers. These cloud providers must provide simple file operations such as uploading, downloading a file and private file sharing. We use one of the cloud service providers in order to store the encrypted file and the other one to store the encryption key. This ensures that both cloud providers cannot see the file content by themselves. We assume that two cloud providers do not collude to bring encrypted file and the key together to decrypt the file. Against such a threat, the solution could be extended to using more than two clouds. Alternatively, encryption keys could be stored in a private, internal cloud. Companies can use their enterprise clouds (which does not need to have a high storage capacity) to store keys and a cloud service provider to store encrypted files.

For file sharing, the application shares the encrypted file and the encryption key to the specified user simply by using cloud provider's file sharing features. With this way, only the specific user can access the encrypted file and the encryption key, and can use them to decrypt the shared file. The solution does not require a complex PKC based key exchange protocol.

Authentication of users is managed by cloud service providers which eliminate the need for a PKI structure or an additional server to store user information. However, a

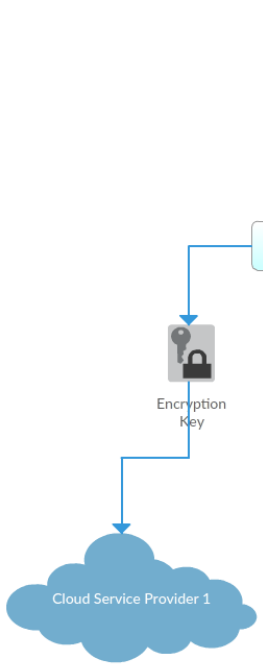


Figure 1. Uploading a file to cloud using TwinCloud.

custom designed authentication method of TwinCloud can be applied.

Our solution does not store user accounts and related key paths to encrypted files on the client side. User accounts are managed by cloud service providers. Key files and encrypted files are uploaded with names that can be matched in order to reach them easily after uploading.

TwinCloud uses symmetric key cryptography for file encryption. The encrypted files are stored in the cloud storage. It deletes the key files and the encrypted files from the local computers temporary folder after uploading and downloading operations are completed. The basic explanation of uploading a file is illustrated in Fig. 1.

In summary, TwinCloud provides a single user account and a uniform user experience to execute all file operations. In the backend, the application manages two different accounts from two separate cloud service providers. While users are uploading their files through the application, files are not stored by our application, rather stored in external cloud drives.

B. Solution Details

In our solution, we choose two cloud service provider; Dropbox and Google Drive as they are preferred the most among cloud users. Moreover, their application program interfaces (APIs) are easy to understand and highly functional.

TwinCloud uses Google Drive to store the encrypted file and Dropbox to store the encryption key. We use Java programming language for implementation. Dropbox API and Google Drive API are used for cloud file operations. Our application uses 256 bit AES with CBC and PKCS7 padding. Encryption keys are randomly generated by Java Key Generator utility. For authentication and authorization, we use OAuth2.0 [17] protocol. Sign up and login operations to Google Drive and Dropbox are also controlled by the application.

There are seven main operations implemented in our application; sign up, login (authentication), upload a file, download a file, delete a file, share a file and unshare a file.

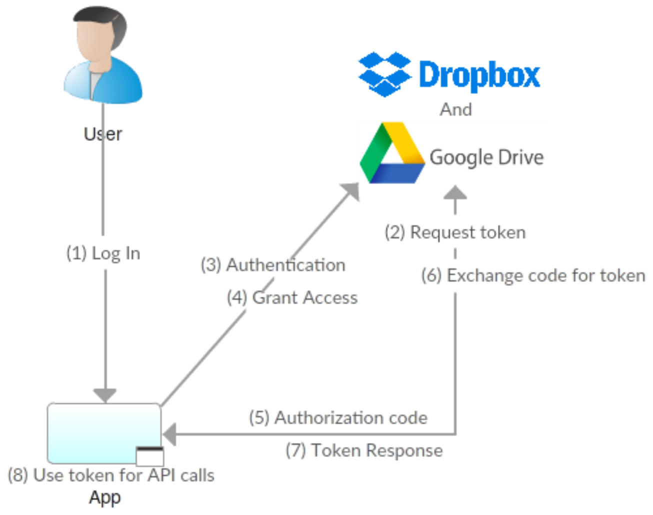


Figure 2. Detailed login flow in TwinCloud.

Fig. 2 shows the login flow. First, user logs in to the TwinCloud by entering his/her username and password (1). TwinCloud, in the backend, performs the login to Dropbox and Google Drive by following these steps: TwinCloud requests an access token from Google Drive and Dropbox separately (2). Dropbox and Google Drive request an authentication. This authentication is performed by the application (3) and the program automatically grants the access (4). Dropbox and Google Drive returns an authorization code (5) which will be used to get the access token. TwinCloud uses the authorization code to exchange the access tokens (6). Finally, Dropbox and Google Drive returns the access tokens (7). The application saves the access tokens in order not to repeat these steps in every login. Authentication and Grant Access operations are implemented using Selenium.

Similar to other cloud applications such as Dropbox and Google, if a user has already an account on TwinCloud and opens the program for the first time, the application automatically starts to download the stored files into the local computer and decrypts them.

Fig. 3 shows the upload, share and download operations.

After login, Alice uploads a file to the TwinCloud (I). The application generates a key (II) and encrypts the file (III). Then, these files are uploaded to Google Drive and Dropbox using APIs as seen in (IV) and (V). After successfully

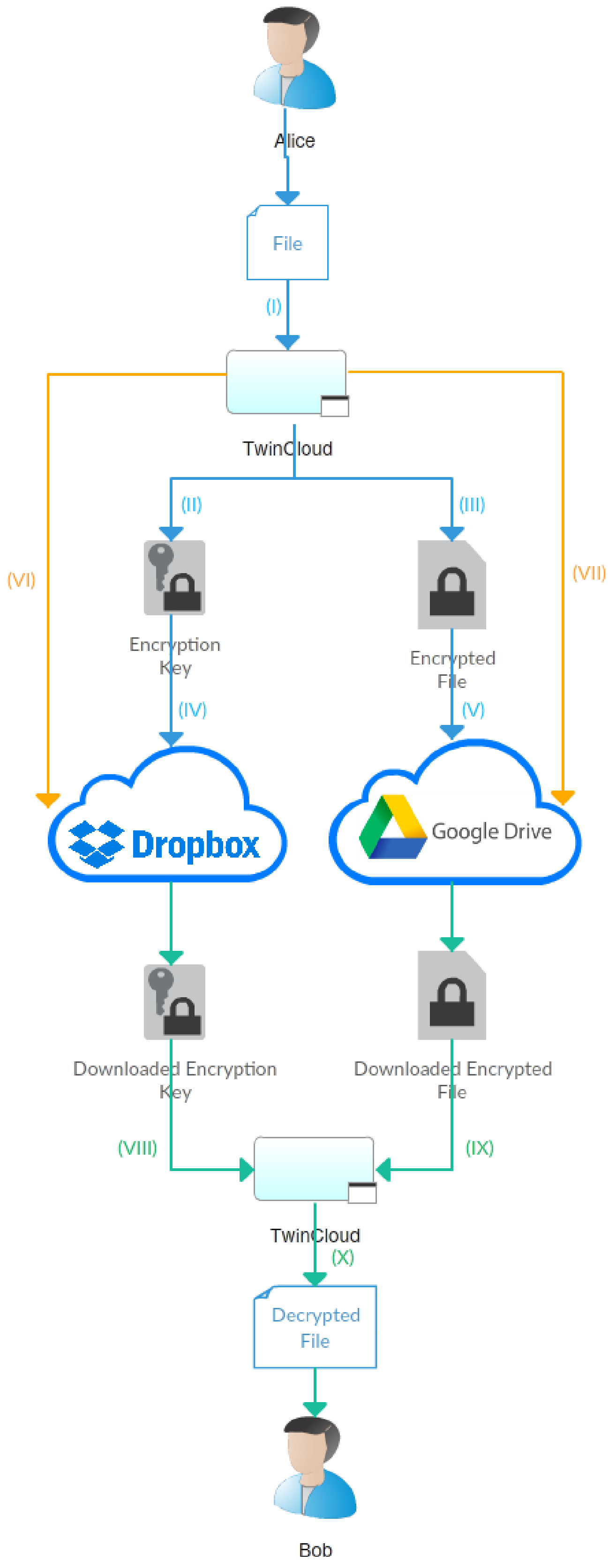


Figure 3. Visualization of the upload, share and download operations in TwinCloud.

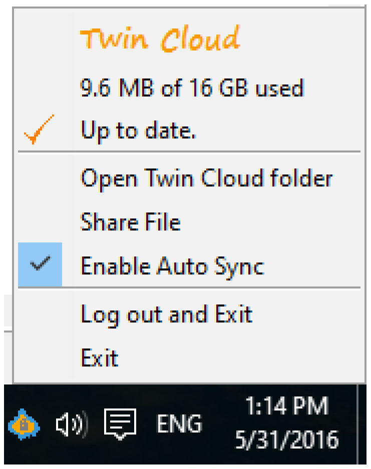
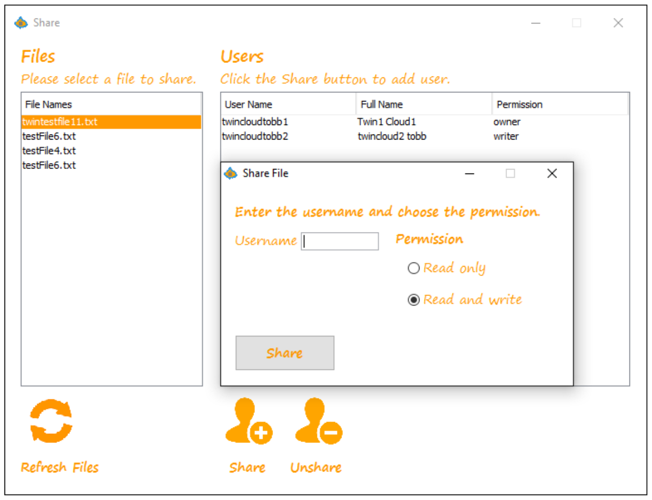


Figure 4. TwinCloud start-bar menu.


Share

Files

Please select a file to share.

File Names
twintestfile11.txt
testFile6.txt
testFile4.txt
testFile6.txt

Users

Click the Share button to add user.

User Name	Full Name	Permission
twincloudtobb1	Twin1 Cloud1	owner
twincloudtobb2	twincloud2 tobb	writer

Share File

Enter the username and choose the permission.

Username:

Permission: Read only [✓ Read and write]

Figure 5. TwinClouds share window.

uploading the file, Alice shares the file with Bob using Dropbox API (VI) and Google Drive API (VII). This will allow Bob to see the shared file from its application. Bob requests to download the file from the application. Our application downloads the encryption key from Dropbox (VIII) and encrypted file from Google Drive (IX). After completing the downloads, the application decrypts the file (X). From now, Bob can also see the content of the shared file.

We implemented TwinCloud in Java. Users can operate TwinCloud from a start-bar menu as shown in Fig. 4 and share their files from the application as shown in Fig. 5. TwinCloud is an open source application and can be found on Github [18].

IV. Comparisons, Extensions And Future Works

Table. I provides a comparison of TwinCloud and other solutions.

	TwinCloud	PKI-Based	Traditional Client-Side Encryption	Server-Side Encryption
Requires a trust to the cloud service provider	✗	✗	✗	✓
Requires a trusted 3rd party	✗	✓	✗	✗
Public key cryptography	✗	✓	✗	✗
Symmetric key cryptography	✓	✗	✗	✓
Easy key management and key exchange	✓	✗	✗	✗
Private sharing	✓	✓	✗	✓
Secret-URL sharing	✓	✓	✗	✓
Integrity check	✓	✓	✗	✓

Table I
COMPARISON OF TWINCLOUD TO OTHER SOLUTIONS.

To hide the real file names from cloud providers, we encrypt the file names while uploading to Google Drive and Dropbox. To do this, TwinCloud generates two encryption keys. We store the first one in Google Drive and use to encrypt the file names on Dropbox. The second key is stored in Dropbox and used to encrypt the file names on Google Drive. Moreover, to protect the integrity of a file, TwinCloud computes a Message Authentication Code (MAC) of the original file and uploads the result to Dropbox. The key used to calculate the MAC is stored in Google Drive.

For future work, for information-theoretical level of security instead of AES, one-time pads can be used to encrypt the files. We note that long key sizes are less of a problem and they can be easily stored in the cloud and shared with other users using our solution. Furthermore, our solution can easily be adapted to more than two cloud providers. For instance, two of the three cloud providers can be used to store the encryption key and the other one can be used to store the encrypted file where both keys are needed to decrypt the file.

V. CONCLUSION

In this paper, we introduced TwinCloud to provide an easy-to-use and secure user experience for cloud sharing. We described a novel approach that uses two or more cloud service providers to securely store and share files without explicit key management. TwinCloud provides security by using one cloud provider to store the encrypted file and the other one to store the encryption key. It executes all the necessary file operation in the back-end and does not ask cloud users for complex operations. Since, TwinCloud does not need to store encryption keys, encrypted files and user information, it is a lightweight solution. TwinCloud's usability and perceived security were evaluated and compared with Tresorit by a usability study which could be found in a longer version of this paper [19]. TwinCloud is an easily applicable security solution in scenarios where it is assumed that cloud providers do not collude with each other.

REFERENCES

[1] C. Louie, Have you enabled two-step verification?, Dropbox Blog, October 2014, Available: https://blogs.dropbox.com/dropbox/2014/10/have-you-enabled-two-step-verification/ (last access June 21, 2016).

[2] Dropbox, How secure is Dropbox?, Dropbox Help Center, Available: https://www.dropbox.com/en/help/27 (last access June 21, 2016).

[3] D. Sheng, D. Kondo, and F. Cappello, Characterizing cloud applications on a Google data center, IEEE 42nd International Conference on Parallel Processing. Lyon, pp. 468-473, October 2013.

[4] Google, Google terms of service, April 2014, Available: https://www.google.com/intl/en/policies/terms/ (last access June 21, 2016).

[5] E. Macaskill, and G. Dance, NSA Files: Decoded, November 2013, Available: http://www.theguardian.com/world/interactive/2013/nov/01/snowden-nsa-files-surveillance-revelations-decoded (last access June 21, 2016).

[6] J. Kincaid, Google confirms that it fired engineer for breaking internal privacy policies, September 2010, Available: http://techcrunch.com/2010/09/14/google-engineer-spying-fired/ Chu (last access June 21, 2016).

[7] E. Duarte, F. Pinheiro, A. Zquete, and H. Gomes, Secure and trustworthy file sharing over cloud storage using eID tokens, OID conference, 2014.

[8] X. Chun Yin, Z. Guang Liu, and H. Jae Lee, An Efficient and secured data storage scheme in cloud computing using ECC-based PKI, Advanced Communication Technology (ICACT) IEEE, 2014.

[9] Boxcryptor, Available: https://www.boxcryptor.com/en (last access June 21, 2016).

[10] Tresorit, Available: https://tresorit.com/ (last access June 21, 2016).

[11] Whitten, Alma, and J. Doug Tygar. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. Usenix Security. 1999.

[12] C. Kang Chu, W. Tao Zhu, J. Han, J. Liu, J. Xu, and J. Zhou, Security concerns in popular cloud storage services, IEEE Pervasive Computing, October 2014.

[13] Boxcryptor, Technical Overview, Available: https://www.boxcryptor.com/en/technical-overview (last access June 21, 2016).

[14] P. Gharjale, and P. Mohod, Efficient public key cryptosystem for scalable data sharing in cloud storage, Computation of Power, Energy Information and Communication (ICCPEIC), April 2015.

[15] C. Kang Chu, S. Chow, W. Guey Tzeng, J. Zhou, and R. Deng, Key-aggregate cryptosystem for scalable data sharing in cloud storage, IEEE Transactions on Parallel and Distributed Systems Volume 25, pp. 468-477, April 2013.

[16] D. Wilson, and G. Ateniese, To share or not to share in client-side encrypted clouds, Information Security Lecture Notes in Computer Science Volume 8783, 2014, pp. 401-41.

[17] [RFC6749] D. Hardt, "The OAuth 2.0 authorization framework", RFC 6749, October 2012.

[18] D. Deniz Yavuz and S. Gurkan, TwinCloud, 2016, GitHub repository, https://github.com/DenizYavuz/TwinCloud

[19] K. Bicakci, D.D. Yavuz, and S. Gurkan, TwinCloud: A Client-Side Encryption Solution for Secure Sharing on Clouds Without Explicit Key Management, arXiv preprint arXiv:1606.04705, 2016.