CloudShare: Towards a Cost-Efficient and Privacy-Preserving Alliance Cloud Using Permissioned Blockchains

Yandong Li[1], Liehuang Zhu[1], Meng Shen[1(✉)], Feng Gao[1], Baokun Zheng[1,2], Xiaojiang Du[3], Sheng Liu[4], and Shu Yin[4]

[1] Beijing Institute of Technology, Beijing, China
{leeyandong,liehuangz,shenmeng}@bit.edu.cn, gaofengbit@foxmail.com, zhengbk168@163.com
[2] China University of Political Science and Law, Beijing, China
[3] Temple University, Philadelphia, USA
dxj@ieee.org
[4] Union Mobile Financial Technology Co., Ltd., Beijing, China
{liusheng,yinshu}@umfintech.com

Abstract. Data explosion has raised a scalability challenge to cloud storage management, while spinning disk capacity growth rates will continue to slow down. Major data holders such as cloud storage providers with a heavy reliance on disk as a storage medium will need to orchestrate multiple kinds of storage to better manage their relentless data growth.

In this paper, we first explore the scenario that multiple clouds are driven by interests to make the storage resources efficiently allocated without requiring a trusted third party, and then propose a novel model, called CloudShare, to enable multi-clouds to carry out a transparent encrypted data deduplication among cross-users via blockchain. Our scheme significantly reduces the storage costs of each cloud, and saves the upload bandwidth of users, while ensuring data confidentiality and consistency. We demonstrate via simulations on a realistic datasets that CloudShare achieves both the effectiveness and the efficiency.

Keywords: Cloud storage · Blockchain · De-duplication
Cost-efficient · Privacy-preserving · Sharing economy

1 Introduction

The advent of cloud storage motivates organizations and ordinary people to outsource data storage to third-party cloud provides. Nevertheless, the fast growth of data volumes in cloud leads to a dramatically increased demand for storage space and upload bandwidth [1]. At the same time, Waldrop [2] predicts the end of *Moore's law* that has powered the information-technology revolution for the

J. Hu et al. (Eds.): MONAMI 2017, LNICST 235, pp. 339–352, 2018.
https://doi.org/10.1007/978-3-319-90775-8_27

past 50 years, which brings new challenges to cloud storage providers ($CSPs$) to maintain a low cost in a sustainable manner.

The cost of $CSPs$ is growing as data grows, whereas cloud storage prices have plummeted over the past few years on account of an ongoing price war among storage service providers [3]. It is estimated that 36% of all data have be stored in the cloud by 2016, compared to just 7% in 2013 due to falling online storage prices [3]. The decrease in barriers to adopt cloud storage further aggravates the storage costs of each CSP. Once the provider runs out of capacity, it has to make another sizable investment (in storage media, or network switches and other infrastructure), forming a relatively high marginal cost. Thus, how to reduce the ongoing storage and maintainability cost and re-evaluate the cloud storage strategies will become an urgent thing in the coming future.

Data deduplication is considered as a promising technology to address the challenges of scalability and complexity of enterprise networks and data centers [4]. Network storage service providers may deduplicate by keeping only one or a few copies for each file to reduce unnecessary redundant copies and generating a link to the file for users asking to store the file. It offers secondary cost saving in power and cooling achieved by reducing the number of disk spindles. These savings also translate to lower fees for customers. A constructive concept for remote data storage is cross-user deduplication, in which if two clients upload the same file, the storage server detects the deduplication and stores only a singe copy. This kind of deduplication will save both the storage capacity as well as the communication bandwidth. According to a survey by IDC [5], 75% of today's data are duplicated copies, thus deduplication achieves high storage savings. However, sensitive data usually be encrypted for privacy before outsourcing, which makes it at odds to conduct a cross-user deduplication, for the same file may have different encrypted copies in the cloud. Although several effective approaches have been put forward in view of this situation, these solutions are limited to a single cloud storage service. Existing solutions are thoughtless of the fact that popular movies, some applications, backup images, etc., tend to be distributed in different cloud storage, but as far as we know, there are currently no preferable solutions to enable multi-clouds to carry out a transparent encrypted data deduplication among cross-users. Despite the dilemma of this issue, there is also no transparent relation between their benefits and the storage offered by different $CSPs$ for them to readily conduct such a co-operation data deduplication.

Inspired by the concept of the *Sharing Economy* [6], we first imagine such a stirring scenario to combine the storage of every cloud to make a *huge alliance cloud* without trusting a center authority, which is supported by the novel technology, called blockchain [7]. The blockchain technology brings us a way that $CSPs$ can maintain a tamper-resistant ledger, shared by the participating members, without the need for a trusted third party, potentially making the $CSPs$ a possible collaboration in an unprecedented way. In this way, different cloud resources can obtain an effective integration and allocation. Roughly speaking, in our scenario, each cloud serves its own users, but when the client C_1 of the CSP_1 wants to upload a file existing in another cloud, such as CSP_2. The CSP_1

will learn there is another copy in CSP_2 through blockchain and then only record the ownership of the file instead of storing it. That is, each cloud does'not need to store all the files, but presents to have all the files. When the client C_1 downloads the file, it is implicitly download from CSP_2, and the CSP_1 only need to pay the CSP_2 a very small fee or anything else. For the cloud who store the file, the cloud increased storage resource utilization and can obtain income from other clouds; For the cloud who doesn't store the file, this scheme reduce its storage overhead. For users, due to each file will have a greater probability to appear in the cloud, they do not need to upload the entire file, which saves both bandwidth and uploading time. In this paper, we will try to make a deduplication over encrypted files in such a scenario.

To summarize, we make the following key contributions:

- For the first time, we explore the scenario that ***multiple clouds are driven by the interests to work together to achieve an efficient allocation for cloud resource without requiring a trusted third party***, effectively slowing down the gap between data generation speed and storage changes, which is beneficial for the limited storage resources now, and then describe the feasibility of the scene.
- To the best of our knowledge, we first propose a novel model, called ***CloudShare***, to enable multi-clouds to carry out a transparent encrypted data deduplication among cross-users by adopting a blockchain (ledger) to record the existence and ownership of the file, which is a ***privacy-preserving cross-user data deduplication scheme supporting client-side encryption without requiring any additional independent servers***. Our scheme significantly ***reduces the storage costs of each cloud, and saves the upload bandwidth of users, while ensuring data confidentiality and consistency***.
- We demonstrate via simulations that CloudShare achieves both the effectiveness and the efficiency.

Roadmap: The remainder of this paper is organized as follows. The background and related work are discussed in Sect. 2, followed by the system model, and the threat model discussed in Sect. 3. Section 4 introduces the preliminaries and definitions. We present details of our CloudShare scheme and discuss its security in Sects. 5 and 6, respectively. The design of experiments and results are demonstrated and discussed in Sect. 7. Finally, we present our conclusions and perspectives for future work in Sect. 8.

2 Background and Related Work

2.1 Blockchain

Blockchain, commonly known as an emerging technology underpinning bitcoin, was first conceptualized by Nakamoto in [8]. It enables an evolving set of parties to maintain a safe, permanent, and tamperproof ledger of transactions without

a central authority, making its potential applications go well beyond enabling digital currencies [9]. Blockchains can be either public, allowing anybody to use them (e.g.,bitcoin) or permissioned, creating a closed group of known participants working to provide an immutable ledger that captures the existence of digital facts in a given moment in time in a non-repudiable manner [10]. We can think blockchain a distributed database contains a list of ordered and relevant records called blocks, each comprising a timestamp and a link to a previous block, as well as a set of transactions. Once a participant wants to add a transaction to the ledger, the transaction data will be verified by other participants in the network using cryptographic algorithms. These transactions are broadcast and recorded by each participant in the network, and are finally recorded in a block by a consensus algorithm (e.g. POW [11], PBFT [12]). Once a block is collectively accepted, it is practically impossible to be changed, which make transactions are immutable, trusted, and auditable.

2.2 Multi-cloud Storage

Multi-cloud storage is currently an emerging technique using a series of clouds to provide data storage service for clients. Most existing multi-cloud storage systems [13,14] mainly focus on data reliability regarding cloud failures and vendor lock-ins. MetaStorage [15] and SPANStore [16] provide both integrity and availability guarantees by replicating data across multiple clouds using quorum techniques. However they don't address confidentiality, which is later achieved by Hybris [17] by dispersing encrypted data over multiple public clouds via erasure coding and keeping secret keys in a private cloud. CDstore [18] builds on an augmented secret sharing scheme called convergent dispersal to achieve both bandwidth and storage savings, whereas it does not address consistency issues due to concurrent updates as mentioned in [19].

2.3 Data Deduplication Security

The demand for data privacy and security is increasing recently [20–26]. Sensitive data usually have to be encrypted before sending to servers, which makes the storage server can neither recognize that the files are identical or coalesce the encrypted files into the space of a single file. Convergent encryption [27] provides confidentiality guarantees for deduplication storage, and has been implemented and experimented in various storage systems. Bellare et al. [28] generalize convergent encryption into Message-locked encryption (MLE) and provide formal definitions of privacy and tag consistency. The same authors also prototype a server-aided MLE system DupLESS [29], which address the applied aspects of encrypted duplication storage. Liu et al. [30] introduce a single-server cross-user deduplication scheme with client-side encryption using a password-authenticated key exchange protocol for MLE key generation. ClearBox [31] enables clients to verify the effective storage space that their data occupies after deduplication. To the best of our knowledge, we are the first to think about the co-operation data deduplication of multiple clouds, and its security.

3 A Hierarchical Framework for CloudShare

3.1 System Model

CloudShare is designed for multiple $CSPs$ to serve their own group of users, simultaneously, they could readily conduct such a cooperation to save storage cost and obtain some extra earnings from other cloud without reliance on a trusted center party. As shown in Fig. 1, a CloudShare framework can be abstracted as two layers:

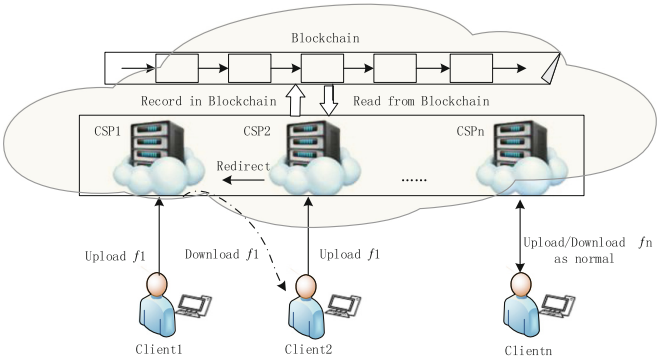

Fig. 1. A blockchain-based hierarchical architecture for CloudShare

1. the upper layer leverages the blockchain-based trading pattern among $CSPs$ (i.e., $CSP_1, CSP_2, ..., CSP_n$). Since $CSPs$ are the different market entities, we consider each CSP interact with such a blockchain, which can be thought as a conceptual party (in reality decentralized) that can be trusted for correctness and availability. Such a blockchain provides a powerful abstraction for the design of distributed protocols. The cloud can write contents into the blockchain and read the contents from it. Once a block is collectively accepted, it is practically impossible to change it or remove it, which is guaranteed by the nature of the blockchain.
2. the bottom layer consists of the $CSPs$ and their corresponding users; Users of each CSP run the CloudShare client to store their data in their own cloud but may access its data in multiple clouds over the Internet. Each cloud can choose to record the unique identifier of the file (i.e., a cryptographic hash of the content of the file) and its associated information into the blockchain so that other clouds can find the file through it. Once the cloud put the unique identifier of the file into the blockchain. It means it should be responsible for the availability of the files. $CSPs$ run the CloudShare server to serve their own users like before. Only one or a few instances of the file is saved and subsequent copies are replaced with a "stub" that points to the original file maybe in another CSP. Meanwhile, each cloud may response the requests from the users of another CSP to download a file it holds and receive a small fee or anything else for other clearing agreement from that CSP.

Taking into account the privacy enforcement for current cloud storage, it is desired to achieve deduplication and encryption simultaneously. The specific approach will be detailed in Sect. 5.

Remark 1. Why we choose blockchain to achieve our idea?

In reality, $CSPs$ can not fully trust each other in the business environment. The use of blockchain has the following advantages, which are difficult to be satisfied by any other mechanism simultaneously.

- The blockchain protocol is a decentralized protocol framework without requiring a trusted central authority to reach a consensus.
- The blocks can be quickly synchronized across distributed nodes which is very suited for $CSPs$ to synchronized a global view of current files.
- The blockchain is practically impossible to be changed once collectively accepted, thus it is undeniable for $CSPs$ to tamper with the data, after announcing the possession of the file in the blockchain.

Remark 2. What drives $CSPs$ to cooperate?

On the whole, once each CSP are actively involved in the system, it could use the fixed storage to exchange for more storage space, and the files it is originally supposed to store can help it earn some other incomes. In addition, the encrypted data is more willing to be shared for each CSP, which is supported by our scheme. Apparently, cooperation can maximize the profit, so we think the cloud is inclined to cooperate. Of course, some $CSPs$ may strongly store more and more additional data, but the storage capacity of each CSP is not unlimited, which will eventually achieve a dynamic balance state.

3.2 Threat Model

In our CloudShare scheme, we assume the decentralized consensus protocol is secure and the blockchain can be trusted for correctness and availability but not for privacy. We assume data is unpredictable (have high min-entropy) to adversary, not to legitimate clients. We assume the existence of encrypted and authenticated channels(e.g., using SSL/TLS) between the clients and $CSPs$, so as to defend against any eavesdropping activity in the network. We consider the following factors that may impact the security of data stored on cloud servers:

1. An outside adversary plays a role of a client that interacts with $CSPs$, attempting to obtain the ownership of the data without the possession of the whole file.
2. The outside adversary may collude with curious $CSPs$ and get access to the storage or the CSP itself are curious about the real data of their clients to extract some information about clients' data or the keys about the data while following the protocol correctly.
3. Selfish $CSPs$ may potentially misbehave in order to save resources (e.g., deleting or tamper data stored on it, and refused to admit what it had done.), after it announced the possession of the data in the blockchain.

4 Preliminaries and Definitions

Convergent Encryption. Convergent encryption (CE) is a cryptography scheme that produces identical ciphertext files from identical plaintext files, irrespective of their encryption keys. Thus, it can be used to provide data confidentiality in deduplication. Specifically, a data owner derives a convergent key K from an original data copy M and encrypts the data copy with K to get the ciphertext C. Beyond that, the data owner also derives a tag τ for the data copy, such that τ will be used to detect duplicates. Note that the way to produce the tag cannot be used to deduce the convergent key and compromise data confidentiality.

Definition 1 *(Convergent encryption (CE)).* *A convergent encryption scheme can be expressed as the triple* $(KeyGen, Enc, Dec, Tag)$ *such that:*

- $KeyGen(M) \rightarrow K$*, where* $KeyGen()$ *is a cryptographic hash function, taking data* M *as inputs and a convergent key* K *as output.*
- $Enc(K, M) \rightarrow C$*, where Enc is a symmetric encryption algorithm that takes both* K *and* M *as inputs and then outputs a ciphertext* C*.*
- $Dec(K, C) \rightarrow M$*, where* $Dec()$ *is a symmetric decryption algorithm that takes both* C *and* K *as inputs and then outputs the original data copy* M*.*
- $Tag(M) \rightarrow \tau$*, where* Tag *is the tag generation algorithm that takes the original data copy* M *or the ciphertext of* M *under the encryption algorithm Enc, then we get a tag* τ *of the data copy* M*.*

Digital Signature. Digital signature is an identity authentication mechanism which is based on asymmetric cryptography theory. In a signature scheme, a signer first publishes its public key and later signs a message with its private key. Anybody who gets the signed message can utilize the public key of the sender to verify the integrity and nonrepudiation of the message.

Definition 2 *(Digital Signature).* *A signature scheme can be expressed as the triple* $(KeyGen, Sign, Vrfy)$ *such that:*

- $KeyGen(1^k) \rightarrow (PK, SK)$*, where KeyGen is a key generation algorithm taking a security parameter* k *as input and outputting a pair of keys* (PK, SK)*, which are called the public key and the private key, respectively.*
- $Sign(SK, m) \rightarrow \sigma$*, where Sign is a signing algorithm taking a private key* SK *and message* m *as inputs and outputting a signature* σ*.*
- $Vrfy(PK, m, \sigma) \rightarrow b$*, where* $Vrfy$ *is a deterministic verification algorithm, taking a public key* PK*, a message* m*, and a signature* σ *as inputs and a bit* b *as output. When* $b = 1$*, it means the signature* σ *is valid and* $b = 1$ *means invalid.*

5 The Proposed Scheme: CloudShare

5.1 Overview

CloudShare ensures a transparent data deduplication among $CSPs$ without compromising the confidentiality of the stored data. We mainly focused on the definition of the two most important operations in cloud storage: storage and retrieval. We expect to combine the novel blockchain technology and convergent encryption techniques. Specially, blockchain enables $CSPs$ to synchronized the global view of the files each cloud holds and convergent encryption technique allows cross-users to securely make a data deduplication on cyphertexts. Furthermore, in order to fit multiple cloud applications, the scenario requires users to prove possession of data prior to its upload. The scheme makes full use of the basic cryptographic primitives, making it computationally-efficient to support cross-cloud data deduplication.

5.2 Scheme Description

We consider multiple $CSPs$ (i.e., $CSP_1, CSP_2, ..., CSP_n$) adopt a permissioned blockchain donated by B as a distributed database. $CSP_i (i \in [1, n])$ initializes its public/private key pair $(Pub_i, Priv_i)$, and publishes the public key Pub_i to all other $CSPs$ in the blockchain network. Also, it initializes a rapid storage system for storing the tags table $TAB(clients, tag, clouds)$ for efficiency. We assume each public key is authenticated by each other. Similar to existing storage providers, CloudShare supports the following operations: Put, Get. In addition, CloudShare supports Proof of Ownership (POW), which is used to generate a proof of data possession. For simplify, We will choose two of the clouds to conduct a cross-cloud deduplication as an example to describe the details of CloudShare.

Specification of the Put Procedure
The Put protocol is executed between the $CSPs$ and clients who aim to upload a file f. Let $H : \{0,1\}^* \rightarrow \{0,1\}^l$ be a cryptographic hash function, where l represents the token size. Clients initialize a convergent encryption scheme $(KeyGen, Enc, Dec, Tag)$, which will be used to encrypt the data of clients and conduct the secure deduplication in the CSP. To store a data file in CSP(e.g., CSP_1), a client $\mathcal{C}_1$ and his CSP_1 performs the following operations:

1. For a file f to be uploaded, $\mathcal{C}_1$ first generates a hash key $K_f = KeyGen(f)$, instead of a random key, derived from f, where $KeyGen$ is a (optionally salted) hash function H (e.g., SHA-256):

$$K_f = H(f) \tag{1}$$

2. To achieve confidentiality, $\mathcal{C}_1$ encrypts the data file f with key K_f to $f^* = Enc(f, K_f)$, where Enk denotes a symmetric key encryption function (e.g., AES-256). $\mathcal{C}_1$ then saves K_f in the local place and computes a digital fingerprint $\tau = H(f^*)$ of f and save it in the local place.

3. Prior to uploading the file, $\mathcal{C}_1$ sends τ to its CSP_1, which will serve as a practically unique handle to f. Here, we take this process as a POW scheme. It is initialized for the client to prove its knowledge of the file.
4. Upon receiving τ, CSP_1 will compute $\tau^* = H(\tau)$, and check whether there is a τ^* in the B. This process can be divided into two cases according to the query result in B.
   (a) $Case1$: Initial Upload of File f.
   If τ^* does not appear in the B, CSP_1 responds the result to $\mathcal{C}_1$ and asks $\mathcal{C}_1$ to upload the file f^*. CSP_1 then computes $\tau^* = H(H(f^*))$,and record $<\tau^*, PK_1, Sign(SK_1, \tau^*)>$ into the B and record $\mathcal{C}_1$ the owner of file f tagged by τ^* as a tuple $<\mathcal{C}_1, \tau^*, PK_1>$.
   (b) $Case2$: Subsequent upload of file f.
   If the digital fingerprint τ^* already exists, the CSP_1 will construct and maintain a set TAB which contains tuples $<U_F, \tau^*, PK_i>$, where PK_i will be extracted from B as the identity of the CSP_i corresponding to the file referenced by τ^*,U_F is the set of clients that are registered to the file. $\mathcal{C}_i$ will be inserted into U_F. This tuples means $\mathcal{C}_i \in U_F$ has the ownership of the file tagged τ^* existing in CSP_i, which is the cloud storage provider that actually holds the file. Note that this can saves storage space at the server and the bandwidth at both sides.

Specification of the Get Procedure

The specification of the Get Procedure is shown as follows. Specifically, when a client $\mathcal{C}_2$ issues a request to CSP_2 to download a file f referenced by τ^*. It initiates this protocol with its CSP_2:

1. The CSP_2 first query the tuples referenced by τ^* and checks if $\mathcal{C}_2$ has the ownership of file f. If this verification passes, it further verifies where the file is. This process can be divided into two cases according to the query result.
   (a) $Case1$: The file f is stored in the CSP_2 itself.
   If the CSP_2 itself holds the file, it transforms the file to a package denoted by (f^*, τ^*), where f^* and τ^* are the encrypted contents of f and its hash tag, respectively. Then, the client can download the encrypted file f^*.
   (b) $Case2$: The file f is stored in $CSP_i(i! = 2)$.
   If CSP_i(i.e., CSP_1) holds the file, actions are taken after CSP_2 pays to CSP_1 a very small fee or anything else, such that eventually the client can download the encrypted file f^* denoted by (f^*, τ^*) from CSP_1 and decrypt it to f with K_f.
2. Upon receiving the package (f^*, τ^*), $\mathcal{C}_2$ fist compute if $H(H(f^*)) = \tau^*$. If not, request to download the file again. If the equation is established, then $\mathcal{C}_2$ can decrypt it to f with K_f by computing $f = Dec(K_f, f^*)$.

6 Security Analysis

We now focus on potential attack scenarios and possible issues that might arise as stated in the threat model Sect. 3.2. Below, we provide an informal security analysis through the following three aspects.

Firstly, an outside adversary without the whole file can play a role of a client that interacts with $CSPs$ and it want to obtain the ownership of the data. Notice that if the adversary attempts to be the owner of the file f without the whole file, it need to prove its knowledge of the file f by providing $\tau = H(f^*)$ as stated in scheme description of CloudShare in Sect. 5. Since $H()$ is a cryptographic one-way hash function, the adversary can not compute a $\tau = \tau$ when it does not know the f^* or the f. Thus, it could not obtain the ownership of f and download it as well.

Secondly, the adversary may collude with semi-trusted $CSPs$ and get access to the storage or the CSP itself are curious about the contents of the data belonging to their clients. However, the files have been encoded by the convergent encryption in our scheme before being outsourced to the $CSPs$. Thus, encrypted files cannot be reverted if the adversary could not get the secret keys in convergent encryption. According to the security definition and analysis for the confidentiality in [28], no efficient adversary $\mathcal{A}$ has non-negligible advantage to distinguish encryption f^* of unpredictable message f from random strings Υ. That is, $\mathcal{A}$ cannot compromise any private access key or private derivation key for unpredictable files f. Thus, Cloudshare can also achieve the security for data based on a secure convergent encryption scheme if the encryption key is securely kept by the clients.

Finally, the blockchian can be assumed to make transactions secure, trusted, auditable, and immutable. Once a CSP put a transaction $<\tau^*, PK, Sign(SK, \tau^*)>$ into blockchain B, which is collectively accepted by most of the $CSPs$. The transaction will be testified by irreversible evidences as stated in Sect. 2. Since we adopted secure signature scheme, the signature unforgeability is also obvious. If the CSP wants to tamper data stored on it, it requires attacking multiple distributed $CSPs$ simultaneously. The consensus protocol featured by blockchain takes by design into account all the $CSPs$. Therefore, there cannot be any database operation completed without most members being aware of it, which ensures data consistency of their clients.

7 Evaluation

In this section, we are dedicated to present the experimental evaluation of CloudShare on a realistic datasets. In our implementation, we let the security parameter $k = 256$ and use the OpenSSL library for all the basic cryptographic primitives. All the CloudShare clients are implemented in Java and the experiment is conducted on some desktop PCs which is running windows and equipped with Intel Core I7 processor at 2.40 GHz and 4 GB RAM. As for blockchain, we select the fabric [32] as the underlying technology of the blockchain-based settlement system, which is an open source permissioned blockchain technique hosted by the Linux Foundation. Our implementation of $CSPs$ interfaces with aliyun servers, which are equipped with Intel Xeon processor at 2.60 GHz and 8 GB RAM. Each CSP is mapped to a node of the blockchain. For a baseline comparison, we also implemented a data deduplication scheme with CE for the

single cloud and integrated it with aliyun server, in which clients directly interact with the single CSP when storing/fetching their files.

7.1 Deduplication Efficiency

An important metric to measure CloudShare is the deduplication efficiency. We thus conduct a evaluation of our scheme compared with the ordinary data deduplication in a single CSP using convergent encryption. We use four real-word sets of files (390 GB in total) to evaluate the deduplication efficiency. All the files were obtained from four personal computers PC_1, PC_2, PC_3, PC_4 in our lab.

We consider adding a file from a PC to its cloud storage as an "upload request". To generate upload requests, each PC will randomly upload its own files from its dataset to its own cloud as far an possible. In our experiment, we map each dataset to a stream of upload requests by generating the requests in random order, where a file that has m copies generate m upload requests at random time intervals. PC_1, PC_2, PC_3, and PC_4 will generate 96335, 87334, 106655, and 77769 upload requests, of which 90123, 80225, 90775, and 69998 are for distinct files, respectively. We define the deduplication percentage p is:

$$p = (1 - \frac{Numbers\ of\ all\ files\ in\ storage}{Total\ number\ of\ upload\ requests}) * 100\% \tag{2}$$

Figure 2 exhibits the deduplication percentage for both single cloud data deduplication (SCD) and multiple cloud data deduplication (MCD). Note that, it is as expected that MCD has a higher deduplication percentage than SCD for each CSP, due to some daily cooperation and some common media files. In reality, popular files such as movies, applications, backup images tend to be distributed in different cloud storage. That is to say, compared with SCD, CloudShare can improve the deduplication efficiency significantly. This also indicates that our method can enjoys lower communication overhead, for users don't need to upload files existing in other $CSPs$, which becomes more significant when the document set is getting larger.

7.2 Performance Evaluation

Blockchain technology can simplify the settlement process, but it may face a performance bottleneck due to the implementation mechanism. In our evaluation, we use fabric to implement the permissioned blockchain and adopt PBFT as the consensus protocol among four $CSPs$, so that the transaction processing speed and transaction confirmation time can be improved significantly. Note that the deduplication process is related to the multiple clouds and multiple clients, in which $CSPs$ need to synchronized a global view of current files through the blockchain. As data deduplication mainly involves the interaction with blockchain, we choose not to do the actual upload and download but just test its conformation time to respond the concurrent number of operation requests from various clients, and then take targeted modifications to adapt to the actual needs of the CloudShare.

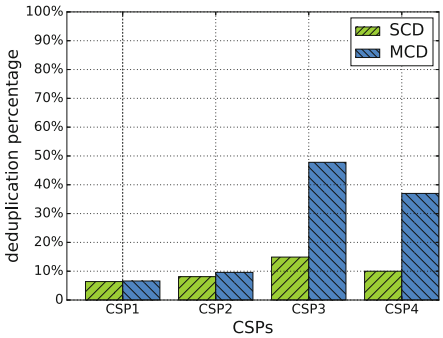

Fig. 2. The deduplication percentage for both SCD and MCD

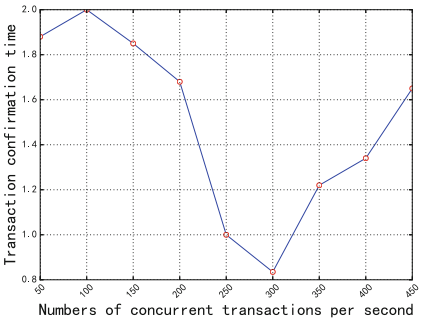

Fig. 3. The relationship between conformation time and the numbers of concurrent transactions per second

Figure 3 depicts the relationship between conformation time and the numbers of concurrent transactions per second. We run the test 30 times and record the mean of 30 time-consuming value as the result. Note that when the volume of concurrent transactions is relatively small, the system has to wait for the predefined batch time to pack a block. In the second stage (100–300), the transaction confirmation time decreases with the increase of concurrent transactions. This is because the number of transactions is reaching the threshold of quantity to pack a blcok in fabric. When the transaction volume exceeds the processing capability, there will be some transactions cannot be confirmed in a timely manner which causes that the transaction confirmation time begins to grow.

It can be seen that our implementation was able to achieve 300 concurrent transactions per second and the transaction confirmation time can be maintained within 2 s on the condition that the number of concurrent transactions is less than the maximum capacity, which satisfies the general requirements. In addition, it is worth noting that when the configuration parameters in fabric are fine tuned, the transaction confirmation time can be reduced to less than 1 s to adapt to the actual needs of the CloudShare.

8 Conclusion and Future Work

We present, CloudShare, a novel scheme via incorporating the blockchain concept into multi-clouds data deduplication as a promising research topic. In CloudShare, a CSP is able to carry out encrypted data deduplication with others in the alliance cooperatively without trusting any trusted third party. More specially, CloudShare greatly saves the cost of storage for multi-clouds and bandwidth for their users, while ensuring data confidentiality and consistency. Extensive security analysis and simulations demonstrate that our proposed scheme satisfies the desired security requirements and guarantees efficiency as well. As part of future work, we plan to investigate how to model the CloudShare via a cooper-

ative game approach to provide an optimization scheme to handle the tradeoff between the storage costs and benefits of each CSP.

Acknowledgments. This work was supported in part by the National Science Foundation of China under Grant 61602039, in part by the Beijing Natural Science Foundation under Grant 4164098, and in part by the BIT-UMF research and development fund.

References

1. Sharma, S.: Expanded cloud plumes hiding big data ecosystem. Future Gener. Comput. Syst. **59**, 63–92 (2016)
2. Waldrop, M.M.: More than moore. Nature **530**(7589), 144 (2016)
3. Paul Rubens. http://www.enterprisestorageforum.com/storage-management/can-cloud-storage-costs-fall-to-zero-1.html
4. Akhila, K., Ganesh, A., Sunitha, C.: A study on deduplication techniques over encrypted data. Procedia Comput. Sci. **87**, 38–43 (2016)
5. International Data Corporation (IDC). https://www.emc.com/collateral/analyst-reports/idc-digital-universe-are-you-ready.pdf
6. Hamari, J., Sjöklint, M., Ukkonen, A.: The sharing economy: why people participate in collaborative consumption. J. Assoc. Inf. Sci. Technol. **67**(9), 2047–2059 (2016)
7. Zheng, Z., Xie, S., Dai, H.N., et al.: Blockchain Challenges and Opportunities: A Survey. Work Pap (2016)
8. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2008)
9. UK Government Chief Scientific Adviser: Distributed ledger technology: beyond block chain. Technical report, UK Government Office of Science (2016)
10. Hardjono, T., Pentland, A.S.: Verifiable Anonymous Identities and Access Control in Permissioned Blockchains (2016). http://connection.mit.edu/wp-content/uploads/sites/29/2014/12/ChainAnchor-Identities-04172016.pdf
11. Gervais, A., Karame, G.O., Wüst, K., et al.: On the security and performance of proof of work blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 3–16. ACM (2016)
12. Castro, M., Liskov, B.: Practical Byzantine fault tolerance. In: OSDI, pp. 173–186 (1999)
13. Kotla, R., Alvisi, L., Dahlin, M.: SafeStore: a durable and practical storage system. In: USENIX Annual Technical Conference, Santa Clara, pp. 129–142 (2007)
14. Hu, Y., Chen, H.C., Lee, P.P., Tang, Y.: NCCloud: applying network coding for the storage repair in a cloud-of-clouds. In: Proceedings of the 10th USENIX Conference on File and Storage Technologies, San Jose, p. 21 (2012)
15. Bermbach, D., Klems, M., Tai, S., et al.: MetaStorage: a federated cloud storage system to manage consistency-latency tradeoffs. In: IEEE International Conference on Cloud Computing, pp. 452–459 (2011)
16. Wu, Z., Butkiewicz, M., Perkins, D., Katz-Bassett, E., Madhyastha, H.V.: SPANStore: cost-effective geo-replicated storage spanning multiple cloud services. In: Proceedings of the 24th ACM Symposium on Operating Systems Principles (SOSP 2013), Farmington, pp. 292–308. ACM (2013)
17. Dobre, D., Viotti, P., Vukolić, M.: Hybris: robust hybrid cloud storage. In: Proceedings of the 2014 ACM Symposium on Cloud Computing (SoCC 2014), Seattle, pp. 1–14 (2014)

18. Li, M., Qin, C., Li, J., Lee, P.P.: CDStore: toward reliable, secure, and cost-efficient cloud storage via convergent dispersal. IEEE Internet Comput. **20**(3), 45–53 (2016)
19. Bessani, A., Correia, M., Quaresma, B., André, F., Sousa, P.: DepSky: dependable and secure storage in a cloud-of-clouds. ACM Trans. Storage (TOS) **9**(4), 12 (2013)
20. Yao, X., Han, X., Du, X., Zhou, X.: A lightweight multicast authentication mechanism for small scale IoT applications. IEEE Sens. J. **13**(10), 3693–3701 (2013)
21. Du, X., Xiao, Y., Guizani, M., Chen, H.H.: An effective key management scheme for heterogeneous sensor networks. Ad Hoc Netw. **5**(1), 24–34 (2007)
22. Du, X., Guizani, M., Xiao, Y., Chen, H.H.: A routing-driven elliptic curve cryptography based key management scheme for heterogeneous sensor networks. IEEE Trans. Wirel. Commun. **8**(3), 1223–1229 (2009)
23. Du, X., Guizani, M., Shayman, M.: Implementation and performance analysis of SNMP on a TLS/TCP base. In: Proceedings of the Seventh IFIP/IEEE International Symposium on Integrated Network Management (IM 2001), Seattle, pp. 453–466. IEEE (2001)
24. Du, X., Chen, H.H.: Security in wireless sensor networks. IEEE Wirel. Commun. **15**(4), 60–66 (2008)
25. Liang, S., Du, X.: Permission-combination-based scheme for Android mobile malware detection. In: Proceedings of IEEE ICC 2014, Sydney, Australia (2014)
26. Shen, M., Ma, B., Zhu, L., Mijumbi, R., Du, X., Hu, J.: Cloud-based approximate constrained shortest distance queries over encrypted graphs with privacy protection. IEEE Trans. Inf. Forensics Secur. **13**(4), 940–953 (2018)
27. Douceur, J.R., Adya, A., Bolosky, W.J., Simon, P., Theimer, M.: Reclaiming space from duplicate files in a serverless distributed file system. In: Proceedings of 22nd International Conference on Distributed Computing Systems, pp. 617–624. IEEE (2002)
28. Bellare, M., Keelveedhi, S., Ristenpart, T.: Message-locked encryption and secure deduplication. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 296–312. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_18
29. Bellare, M., Keelveedhi, S., Ristenpart, T.: DupLESS: Server-Aided Encryption for Deduplicated Storage. IACR Cryptology ePrint Archive 2013, 429 (2013)
30. Liu, J., Asokan, N., Pinkas, B.: Secure deduplication of encrypted data without additional independent servers. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 874–885 (2015)
31. Armknecht, F., Bohli, J.M., Karame, G.O., Youssef, F.: Transparent data deduplication in the cloud. In: ACM SIGSAC Conference on Computer and Communications Security, pp. 886–900. ACM (2015)
32. The Linux Foundation. https://github.com/hyperledger/fabric/