PRIVACY AWARE DATA DEDUPLICATION FORSIDE CHANNEL IN CLOUD STORAGE

R. Anbarasan, Shaki Ahamad M, Mohamed Harsak K, Thahir Ahamed SA
Department of Computer Science and Engineering,
Dhaanish Ahmed Institute of Technology, Coimbatore, Tamilnadu, India

ABSTARCT

Data deduplication is vital technique to compress a data for eradicating duplication of uploading data, and utilized broadly in cloud to minimize the capacity of storage memory and helps in saving bandwidth. To safeguard the truthfulness of delicate data when deduplication process, before outsourcing the data, encryption technique is implemented to encrypt data. To secure data with high efficiency, this paper initially attempts to identify the issue of authenticated data deduplication formally. Unlike conventional deduplication techniques, the users who upload the data to cloud are also considered in verifying duplication beyond the data itself. Discrete new deduplication techniques supporting authenticated duplicate verification ins i d e channel of cloud architecture. Additionally, the user is intimated with IP address of the attacker who tries to obtain or modify the data file uploaded. IP address denotes from where the malicious attack has been done by the attacker. This analysis in security of data demonstrates that scheme proposed is trustworthy in terms of specified definitions in security model. As a proof of concept, a prototype of our proposed authenticated duplicate verification scheme is implemented and carries testing experiments using our prototype. Proposed authenticated duplicate verification scheme experiences negligible overhead when comparing with normal operations.

Keywords: Cloud Storage; Data Privacy; Deduplication Check; Side Channel

I. INTRODUCTION

Cloud computing furnishes clear boundless “virtualized” resources to consumers as services over the Internet, while thrashing platform and execution details. The present cloud specialist providers offer both very accessible storing capability and enormously parallel resources for computing at moderately low expenses. As cloud computing is widespread, storing of data’s to cloud is increasing drastically and shared among users with indicated benefits, which characterize the rights for accessing the data’s stored in cloud. One basic problem

of cloud storage services is to manage the consistency when the magnitude of data increases.

To maintain the managing of data expandable in cloud, deduplication is been a notable system and attracted the attention nowadays. Deduplication is specially designed technique in compressing of a data for disposing the duplication of same files. To enhance utilization of storage and can to organize the quantity of bytes of data that to be sent a technique is followed. Rather than keeping same contents of data with different name, which leads to large memory wastage, deduplication technique is applied to clean out excess information by keeping just a unique copy and mentioning other repetitive information to that duplicate. Checking for duplication can be done in either file or block level. If file level is used, duplication of the similar file is abolished. If it is block level, removes the duplicated block. Although cross-user client-side deduplication check can bring the above benefits, it will also lead to the threat of side channel [4]. According to the traditional deduplication check, if fingerprint matching of a chunk does not exist, it means that there is no such chunk on the serverside. The server-side will feed back to the client-side and request to upload the file chunk. If fingerprint matching is successful, the file chunk does not need to be uploaded. Based on the above observation, the client-side needs to receive the exact feedback from the server-side and decides whether to upload file chunk. However, the users can always obtain the determined existence or inexistence information of the chunk by observing the data traffic transmitted between the

client-side and the server-side. Thus, existence or inexistence response from server-side products the threat of data privacy leakage. and creates a side channel [4]. This privacy leakage can lead to the following potential attacks. In order to identify the existence or the inexistence status of a specific file, an attacker performs a deduplication check by using well-designed file template.Violent file cracking can be seen as the most straightforward privacy leak.

Stealing the file content: Normally, an attacker can check whether a particular file is stored in a cloud stor-age. However, what is more serious is that the attacker might apply this attack to multiple versions of the same file and obtain the secret information of files by brute force. The file T is the target file to be attacked. The attacker has obtained other information about file T except x. In order to steal the secret content of x, the attacker try to use a way of brute force. It uses n files with all possible values of x (x1, x2, ..., xn) to probe deduplication respectively. If only file Fn does not upload the file chunk, the attacker will be very sure that the

secret information is xn. Because the server-side identifies the existence of the unique chunk xn, the content of x is stolen by the attacker. If the number of possible versions of the target file is moderate [5], the success rate of this attack is very high.

Covert Channel: As long as the two parties reach a consensus, a covert channel can bypass the censorship and communicate with each other [12, 22]. It means that file existencestatuscan be used as the medium ofcommunication.

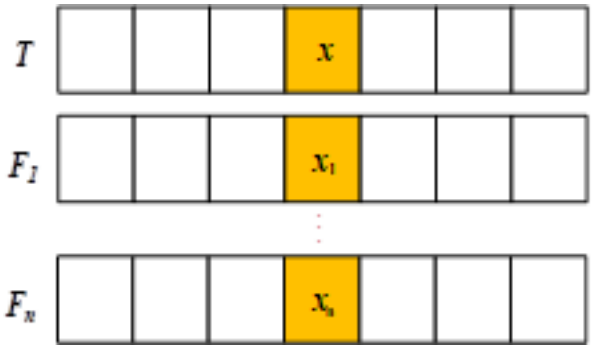

Figure 1: The attack in chunk-level deduplication

The obvious drawback of the traditional deduplication check is that the attacker can continuously use templates of file chunks to violently verify the sensitive information of the file which are stored in server-side, but attacker usually blocks the upload of file chunks. Because the templates of these file chunks can be used repeatedly, thus this is one of the key factors that leakage of user data privacy. However, the server-side does not take effective measures to against this threat. In order to address these challenges, we propose double byte transport protocol (DBTP), a simple yet effective strategy to ensures a balance between privacy security and deduplication check benefit. It achieves the two- side privacy (existence privacy and inexistence privacy). Since deduplication check of single chunk make the user clearly knowing whether the chunk exists on the cloud, the strategy uses auxiliary chunk to perform the deduplication check on double chunks at once and return the

reasonable value. This means that there exist enough room to confuse the attacker's judgment when client-side performs deduplication check. In particular, in order to solve side channel, there is set a list D for recording dirty chunks (file chunks that need to be uploaded but not uploaded

under normal deduplication check). Chunk list H for recording chunks that no longer needs to be checked. It helps save bandwidth and storage overhead. As can be seen from Table 1, compared to existing solutions, the DBTP protocol has its own advantages, such as no redundant parameter configuration, no additional hardware, two-side privacy protection and storage space savings. Table 1: Comparisons between DBTP and other schemes NoArgument NoHardware Privacy Save RT No Yes No No ZEUS Yes Yes No Yes ZEUS+ No Yes Yes No RARE Yes Yes Yes No Mozy No Yes No No Shin Yes No Yes Yes Heen Yes No Yes Yes DBTP Yes Yes Yes Yes The scheme achieves the two-side privacy and maintains the deduplication rate of the original deduplication check. Specifically, if a double chunks combination is confirmed not in the cloud by deduplication check, the client-side will only upload one of the chunks first. The remaining chunk will be combined with the other chunk that is selected from the remaining list chunks. It is different from RARE [5] which implements privacy security with excessive redundant chunks upload. Other similar programs, just like ZEUS [15] can't satisfy the inexistence privacy and RT [4] only offer the inexistence privacy. ZEUS+ [15] is based on ZEUS and RT to enhance the privacy security, but the setting of the random threshold parameter severely reduces the deduplication rate. We conducted a detailed security analysis about the return values of the server-side in the DBTP scheme. It demonstrates that DBTP can effectively resist side channel attack.In addition, this method only involves the interactions between the client-side and server-side and does not need the extra hardware [23]. It is only minimally modified on the original deduplication check technology. The DBTP implementation does not require additional parameter configuration. Most solutions [4, 12] are to ensure privacy security through random threshold parameters. ZEUS+ [15] adopt a random threshold chosen uniformly in a range [2, d] to maintain an inexistence privacy.

II. Background and Related Work

2.1 Deduplication in Cloud Storage

Data deduplication is an effective technique to eliminate the redundant data which results in storage saving directly [8, 20]. For example, as shown in Figure 2, the client-side of

user Aaron first uploads the file fm, then the file fm will be divided into A, B, C, D, E, and F chunks by the dicing algorithm. When the hash values of these chunks are matched, it is found that these chunks are not in the cloud.Therefore, they are all uploaded to the server-side. Later, another client-side of user Beck. wants upload file fn to own cloud folder. After deduplication check,

he or she founds that chunks E, F, and C already exist in the cloud. Thus, the user Beck only needs to upload H, I and J, which are three chunks that did not exist in the cloud. The result is that server-side saves storage and respective bandwidth. It is foreseeable that as the size of users increases, the amount of redundant data is bound to increase more. At that time, the benefits of deduplication check must be considerable. Xia et al. [18] present a P-Dedupe system. It uses pipeline and parallelization techniques to accelerate the deduplication process. In our work, we focus on cross-user client-side data deduplication. In addition, different chunk segmentation algorithms can use different slice sizes for files. For example, Dropbox [3] performs the deduplication check and the file is partitioned to some chunks with a determined value of 4MB size.

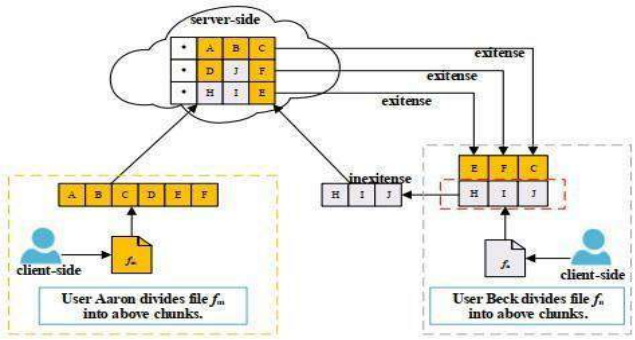

Figure 2: An example of deduplicationcheck

The most mainstream scheme of deduplication check is the cross-user client-side data deduplication. Cross-user means that deduplication check is executed in the

storagespace shared by all users. Compared with deduplication check under single-user, redundant data deletion rate based on cross-user has increased a lot. Client-side means that the deletion of redundant data is performed on the client. The process is that the client-side sends the

hash value of the chunks to the server-side. According to the hash matching results of the server-side, the client-side determines whether the chunks are to be uploaded to the server-side. It is illustrated in Figure 3, where the clientside sends direction request and server-side gives direction response. It is used to tell user whether the chunk need to be uploaded to the cloud.

2.2 Related Work for the Side Channel

Especially, after performing hash matching of file chunk,the cloud needs to deterministically return the existence status.The problem of side channel which is appearing in the traditional deduplication check. It is defined by Harnik et al. [4]. In order to address the problem of side.

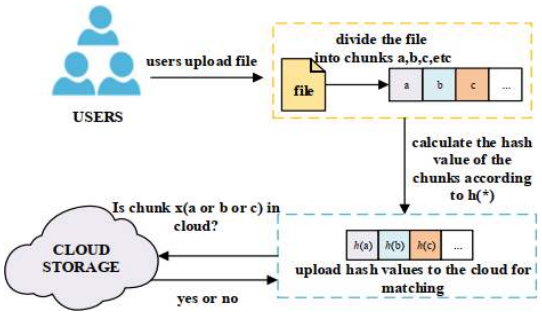

Figure 3: Deterministic response of data deduplication channel

The recommend using a random threshold solution (RT). The server-side assigns a threshold tx which is belongs to [2, d]. Thus, it is only known by the serverside. When a file chunk cx is uploaded and the number of existing copies is greater or equal to tx, the system will perform client-side deduplication. The problem of this scheme is the uncontrollability of the d value. If the d is too large, the number of copies will be too much. If the d is too small, it will lead to lower security. Thus, the choice of this parameter is difficult. However, this random threshold solution also has privacy issues. If the number of chunk copies recorded in the cloud exceeds the specified threshold tx, the deduplication check will expose the privacy of file existence. Compared to the global threshold solution of Harnik, Lee and Choi

[11] adopted a random threshold ti at each uploaded chunk. It is claimed to show stronger privacy than Harnik et al.'s solution. Armknecht et al. [1] recently proved that deduplication thresholds uniformly sampled from [1, B] achieve the optimal defense for the natural privacy

measure. In particular, Wang et al. [9] designed the deduplication thresholds based on a gametheoretic approach. Unfortunately, all of the above proposals are based on the RT category, thus have the same weaknesses.

The idea of Mozy [21] is that only small- size files contain sensitive information, but large- size files such as music and movie are not sensitive. Based on such assumptions, the schemedesigns a threshold x for the file size. If the size

of the file is smaller than x, it is treated as the small file, otherwise it is regarded as the large file. Therefore, when the size of the file is larger than x, the deduplication check will be performed, otherwise it will not. Obviously, this lack of theoretical support, because there is no necessary connection between the size of the file and the importance of sensitive information. Many previous studies focus on deduplication for periodical backup streams [7, 12]. Spatial or temporal locality has been exploited in [17]. Yu et al. [19] propose ZEUS which can achieve weak existence privacy but cannot achieve inexistence privacy. In order to implement two-side privacy, Yu et al. further propose ZEUS+. It is to combine the use of ZEUS and RT.

In addition to above solutions, there exist some researchers recommend using the extra hardware between the client-side and server-side to enhance the privacy of data security. S. Li et al.

[2] propose a secure and efficient client-side encryption deduplication scheme (CSED). This solution introduces a private key server to generate MLE keys to resist brute force attacks. Shin and Kim [12] propose a deduplication protocol. It implements differential privacy check based on an independent server bridging between the client-side and server-side. Similar methods is that Heen et al. [22] considered to set trusted gateway bridges between the users and cloud. Based on the above methods of using additional hardware, the deterministic relationship that

between client-side requests and serverside responses can be broken.

2.3 Design and Implementation

As the methods mentioned above, there exist defects in the processing of channel problems. In this section, the DBTP algorithm is presented. It is used to solve the channel problem and keep the benefits of deduplication check. To ease reading, we summarize major notations, theyare used to construct the DBTP

2.4 The Simple Introduction of DBTP

The biggest problem with the traditional deduplication protocol is that deduplication check with only one chunk at a time. The server-side explicitly returns the state of a chunk's existence. Thus, the attacker can judge whether the chunk has been transmitted by observing network traffic. In DBTP, the basic idea is to use the flag chunk(tag) for auxiliary transmission under different transmission conditions. When the user runs the local client program, the client-side will generate a flag chunk according to the agreement with the cloud. According to the agreement, the cloud defaults the flag chunk already exists. The combination method of double chunks is only [xi , xi+1] or [xi
, tag]. The deduplication check on the other double chunk combination is based on the specific implementation details of the DBTP algorithm. 3.2 Scheme Design in Detail The cloud returns 0 to client indicate that the chunk does not exist in the cloud. If cloud returns 1means the opposite. As you know from the Figure 4, ti denotes the number of chunks that the cloud requires client-side to upload. The t1 represents both chunks c1 and c2 are not in the cloud storage, the t2 represents c1 is not but c2 is in the cloud storage, and so on. Figure 4: The process of the scheme When the uploaded file is split, there are only two types of combinations. One is the combination of ordinary chunks like [x1, x2], neither x1 nor x2 is a tag. The other is the combination of ordinary and tag chunks like [xi , tag]. Deduplication check for a hash list of two chunks at the same time. Obviously, the number of chunks required to be uploaded can only be selected in 0, 1, and 2. First, except for t4, the others cannot equal 0. Otherwise, the chunk will be missing. Thus, DBTP needs to meet t1 6= 0, t2 6= 0, and t3 6= 0. Specifically, if the attacker is interested in chunk c2, he or she can upload two chunks of [c1, n] first. Thus, c1 must exist in the cloud. Then, the attacker can upload [c1, c2]. Obviously, if t3 6= t4 and the state value of c1 both are 1, the existence state of c2 can be easily determined. In this sense, t3 = t4 needs to be satisfied. Similarly, it can be obtained that t2 = t4 must also be satisfied. It is easy to draw the following conclusions that t2 = t3 = t4 and the values must be selected from 1 and 2. Thus, the value of [t1, t2, t3, t4] can be [1,1,1,1], [2,1,1,1], [1,2,2,2] and [2,2,2,2]. Nonetheless, if the result returned by the serverside is [2, 2, 2, 2], it can offer the strongest privacy, but deduplication will be ineffective. However, [2,1,1,1] clearly exposedthe privacy of inexistence which c1 and c2 do not exist in the cloud. Therefore, we can only choose from [1,1,1,1] and [1,2,2,2]. Obviously,

[1,2,2,2] is not conducive to bandwidth savings. Based on the above analysis, the most suitable values of t are [1, 1, 1, 1]. It means that result will be returned to client-side

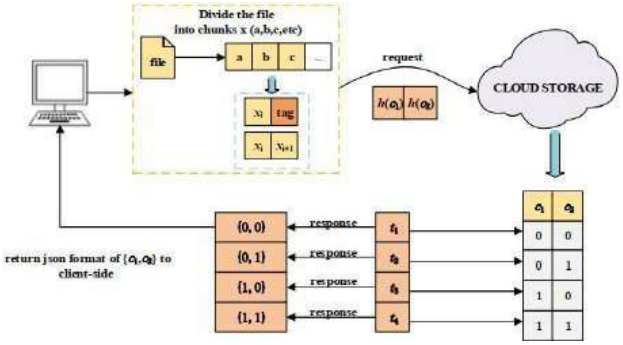

Figure 4: The process of the scheme

III. DBTP Algorithm Description

The algorithm shows the details of DBTP. First,in or- der to prevent the illegal behavior of malicious attackers, the system needs to verify the user's identity information. In the registration stage, the user needs to submit some personal information to the server. After receiving the in- formation, the server will issue a smart card to the client, which contains some security parameters for later authentication [2, 18, 19]. After the registration phase, users can access the server in the authentication phase. Only users with valid smart cards and corresponding passwords can be successfully verified by the server. In that case, users without permission will not be able to access the system for malicious attacks. Besides this, we can also adopt an efficient certificateless conditional privacy preserving authentication scheme [2, 14] or a privacy preserving public auditing scheme [6]. Chunk list H store chunks that have been uploaded. K is a collection of all file chunks.

When the user attempts to upload a file f_x to the cloud. First of all, the file f_x will be divided into chunks by size ϕ. The number of file chunks may be odd or only the last a chunk is left to be checked. Thus, it need use the tag chunk to assist the deduplication check (line 02-03). It should be emphasized that check of tag will

return 1 by default. Specifically, the client-side selects two different chunks, h(ci) and h(cj), to carries out deduplication check (line 05-09). If i is equals to j means that only the last chunk needs to be checked, so it will be matched to the tag chunk for deduplication check (line 05-07). For the two kind of combinations above, client-side performs deduplication check and cloud sends the values of chunk status according to Figure 4 (lines 10-21). Depending on Table 2, client-side receives values of JSON format from cloud and decides which chunk will be uploaded (lines 22-37). Importantly,client-side save the chunk that is uploaded and exist in the cloud already to the H list. When two chunks are combined next time, the chunks in the H list will be ignored. It will speeds up the deduplication check.

Algorithm 1 DBTP

1: Begin

2: client partitions fx into chunks c1,...,cn.3: create a chunk tag = cn+1

4: while i,j ∈ K and i,j ∈/ H do

5: random selection of two chunks [ci ,cj] and i 6= j.6: if i == j then

7: h(cj) = h(tag).8: end if

9: client performs deduplication on [h(ci) ,h(cj)].

10: **if** c_i and c_j not in cloud **then**

11: cloud responses [0,0] according to Figure 4.

12: **end if**

13: **if** c_i not in cloud and c_j in cloud **then**

14: cloud responses [0,1] according to Figure 4.

15: **end if**

16: **if** c_i in cloud and c_j not in cloud **then**

17: cloud responses [1,0] according to Figure 4.

18: **end if**

19: **if** c_i in cloud and c_j in cloud **then**

20: cloud responses [1,1] according to Figure 4.

21: **end if**

22: **if** client receives $[c_i, c_j]$=[0,0] **then**

client uploads c_i or c_j for Table2.
    $H = H \cup c_i$ or c_j
end if
if client receives $[c_i, c_j]=[0,1]$ **then**
    client uploads c_i for Table 2.
    $H \neq H \cup c_i$ and c_j
end if
if client receives $[c_i, c_j]=[1,0]$ **then**
    client uploads c_j for Table 2.
    $H = H \cup c_i$ and c_j
end if
if client receives $[c_i, c_j]=[1,1]$ **then**
    client uploads c_i or c_j for Table 2

IV. Performance Evaluation

4.1 Experiment Settings

Based on Linux system, we have built a preliminary programming and testing environment. The scheme is tested on centos- release-6-8 on Intel(R) Xeon(R) CPU E5-2

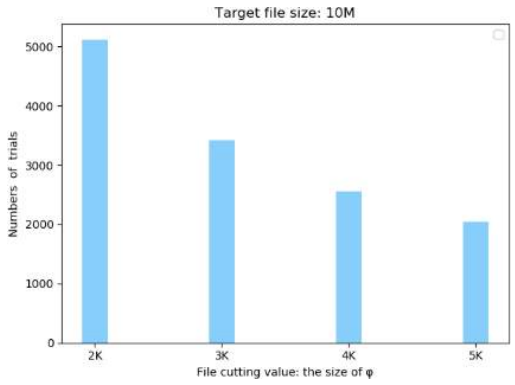

Figure 5: Target file size(10M)

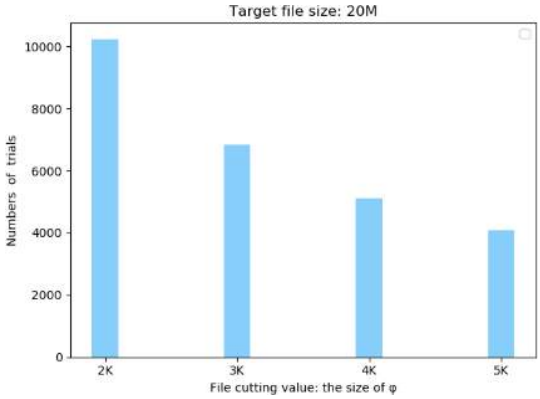

Figure 6: Target file size(20M)

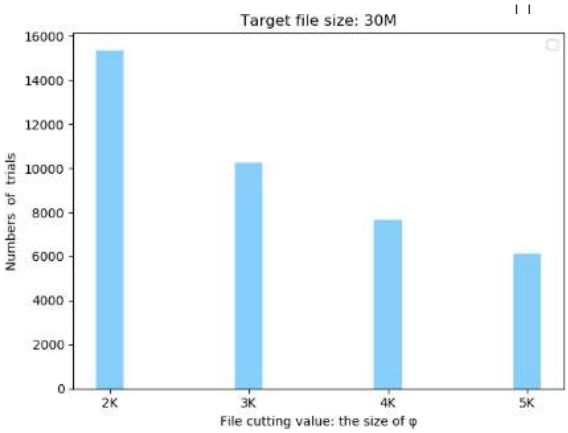

Figure 7: Target file size(30M)

4.2 Experiment Results

DBTP can realize two-side privacy to avoid side channel attack. The privacy experiment uses 10M, 20M and 30M files as the target test object. The cut size ϕ of each file is set to 2K, 3K, 4K, 5K respectively. The experimental results are shown. Deduplication ratio is an important indicator to measure the efficiency of deduplication check. Results are shown in Figures 8, 9 and 10. The deduplication ratio is defined as follows $\alpha = A/S = (S-B)/S$. The formula is explained as follows, α indicates the deduplication rate, and A represents the deleted file size for performing deduplication check operation. The value of A is equal to the difference between S and B, where S represents the total size of the file thatwas not subjected to the deduplication process at the time of uploading, and B represents the file size that has been uploaded to the cloud. The

deduplication rate is not only affected by thededuplication strategy, but also by chunk size.

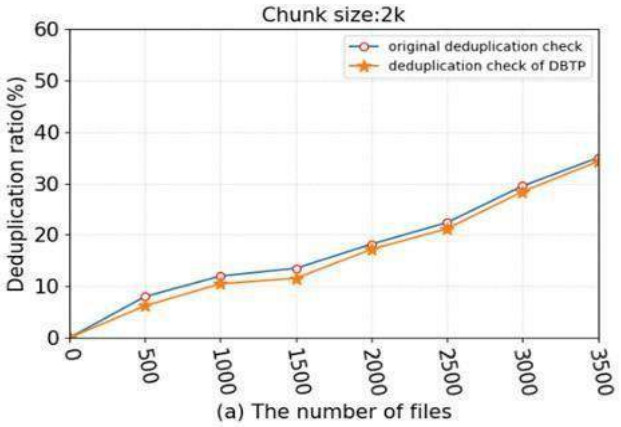

Figure 8: Deduplication ratiocomparison(2k)

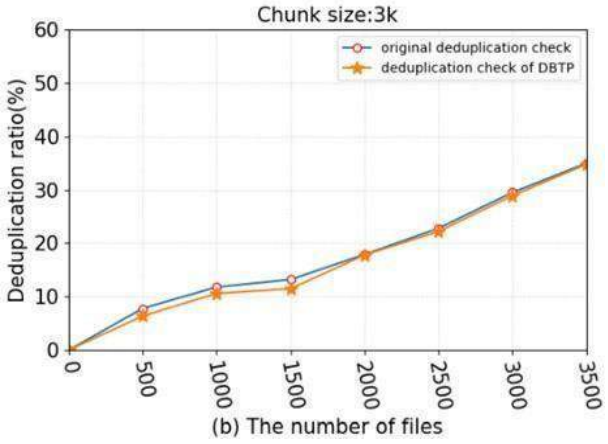

Figure 9: Deduplication ratiocomparison(3k)

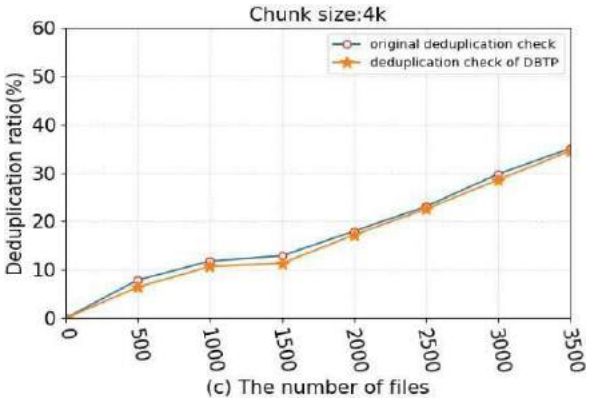

Figure 10: Deduplication ratiocomparison(4k)

V. Result Analysis

In this section, we have completed two experiments. The former is the experiment of DBTP security privacy for avoiding side channel attack and the latter is a compar- ison experiment between DBTP and the original schema deduplication rate. Based on Figures 5, 6, and 7, we ex- amine files of sizes 10M, 20M, and 30M, respectively. For example, the target file T, the attacker does not know about a small part of the file, but this part is the privacy part that the attacker is more interested in. Now, we use a computer B to play an attacker to upload different file templates. These templates only modify the imaginary privacy part. The number of probes for each file upload is shown on the y-axis. Through traffic analysis, whether it is a target file of size 10M, 20M or 30M, when violent trials are cracked on them, file blocks are uploaded every time. Because the attacker cannot observe the transmis- sion of zero traffic, it is

impossible to judge whether the modified data part matches the target file. Based on the above analysis, the DBTP protocol protects the privacy of data.From the Figures 8, 9 and 10, compared the dedupli- cation ratios of the two methods. It is obvious that the original deduplication check has relatively higher dedu- plication ratio compared to the DBTP method. Because the original deduplication check finds the maximum op- portunity to eliminate the redundant chunk. Fortunately, deduplication check of DBTP guarantees the two-side pri- vacy of data, but the deduplication ratio is only a little reduced, and the gap between them is within an accept- able range. In general, DBTP precisely guarantees a good balance between deduplication efficiency and data privacy protection.

VI. Conclusions

Although cloud storage service providers have widely adopted cross-user client-side deduplication check to re- duce redundant data and communication costs, it leaks the privacy of the chunk existence or inexistence status, resulting in more threats like

side channel. In this pa- per, we propose a solution, DBTP, based on tag chunk for auxiliary transmission. This scheme leaks zero-knowledge for side

inexistence status in- formation from deduplication check. DBTP implements a stronger two-side privacy and performance guarantee based on minimal modification of the ordinary deduplica- tion mechanism.

VII. Acknowledgments

This work was supported by the National Key Research and Development Program of China (No.2018YFB0704400).

References

[1] A. Chiniah, J. A. D. Dhora, and C. J. Sandooram, "Erasure-coded network backup system (ECNBS)," in *International Conference on Information, Com- munication and Computing Technology*, pp. 35–43, 2017.

[2] Z. Guo, "Cryptanalysis of a certificateless condi- tional privacy-preserving authentication scheme for wireless body area networks," *International Journal of Electronics andInformation Engineering*, vol. 11, no. 1, pp. 1– 8, 2019.

[3] Z. Han, W. Xia, Y. Hu, D. Feng, Y. Zhang, Y. Zhou, M. Fu, and L. Gu, "Dec: An efficient deduplication-enhanced compression approach," in *IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS'16)*, pp. 519–526, 2016.

[4] D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Side channels in cloud services: Deduplication in cloud storage," *IEEESecurity & Privacy*, vol. 8, no. 6, pp. 40–47, 2010.

[5] S. Lee and D. Choi, "Privacy-preserving cross- user source-based data deduplication in cloud stor- age," in *International Conference on ICT Conver- gence (ICTC'12)*, pp. 329– 330, 2012.

[6] C. Li and Z. Liu, "A secure privacy-preserving cloud auditing scheme with data deduplication," *International Journal of Network Security*, vol 21, no. 2, pp. 199–210, 2019.

[7] S. Li, C. Xu, and Y. Zhang, "CSED: Client-side encrypted deduplication scheme based on proofs of ownership for cloud storage," *Journal of Information Security and Applications*,

[8] D. Meister and A. Brinkmann, "Multi-level compar- ison of data deduplication in a backup scenario," in *Proceedings of SYSTOR: The Israeli Experimental Systems Conference*, pp. 8, 2009.

[9] D. T. Meyer and W. J. Bolosky, "A study of prac- tical deduplication," *ACM Transactions on Storage (TOS'12)*, vol. 7, no. 4, pp. 14, 2012.

[10] J. Paulo and J. Pereira, "A survey and classification of storage deduplication systems," *ACM Computing Surveys(CSUR'14)*, vol. 47, no. 1, pp. 11, 2014.

[11] Z. Pooranian, K. C. Chen, C. Mu Yu, and M. Conti, "Rare: Defeating side channels based on data-deduplication in cloud storage," in *IEEE IN- FOCOM IEEE Conference on Computer Communications Workshops (INFOCOMWKSHPS'18)*, pp.444–449, 2018.