

Enhanced Deduplication Protocol for Side Channel in Cloud Storages

Jie Ouyang¹, Huiran Zhang^{1,2,3}, Hongqing Hu¹, Xiao Wei^{1,2,3}, and Dongbo Dai¹

(Corresponding author: Dongbo Dai)

School Computer Engineering and Science, Shanghai University, Shanghai 200444, China¹
Shanghai Institute for Advanced Communication and Data Science, Shanghai 200444, China²

Materials Genetics Institute, Shanghai University, Shanghai 200444, China³

(Email: dbdai@shu.edu.cn)

(Received July 31, 2019; Revised and Accepted Dec. 3, 2019; First Online Feb. 3, 2020)

Abstract

Cloud storage usually adopt client-based deduplication, which can achieve considerable savings in both storage and bandwidth. However, an attacker can carry out a steal-file-content (SFC) attack, exposing data privacy. In this paper, a simple yet effective scheme, called double bytes transport protocol (DBTP), is proposed. In this scheme, the client-side requests deduplication checks of the double chunks simultaneously, and the server-side receives the deduplication request and response with a reasonable value. The result demonstrates that DBTP can significantly mitigate the side channel's risk while maintaining the high bandwidth efficiency of deduplication.

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

1 Introduction

As a convenient and popular service model, cloud storage services enable users to store and access various resources in the cloud on demand. Major cloud storage providers offer multiple file types of storage. According to a report of an International Data Corporation (IDC), the total amount of digital data in global data centers is exploding rapidly [16]. According to the latest information, the volume of global digital data created and replicated is up to 33 Zettabytes in 2017, while with the advent of the era of big data, it will exceed 175 Zettabytes in 2025. IDC analysis also shows that 49% of the data will be stored in a public cloud environment and close to 75% data has a copy. This similar phenomenon also exists in the research report of Microsoft Research Institute [10]. Based on the above surveys, a large amount of redundant data will storage in the cloud. If the cloud storage services cannot process these redundant data, it will cause waste of cloud storage space and increase network bandwidth when users upload files to cloud storage.

Nowadays, there are two main ways to implement the

deduplication check. They are service-side deduplication check and client-side deduplication check respectively. In the former, the users must upload the data to the server-side first, then server-side deletes the redundant data again. This approach can reduce storage overhead, but cannot save bandwidth. In the latter, client-side divide the file into multiple chunks and send chunks hash signatures to cloud and check for the existence or inexistence status of chunks. It can not only delete redundant data but also save bandwidth. Therefore, the commercial cloud storage services mostly adopt the cross-user client-side deduplication check to maintain a specified number of copies of files [13]. It can achieve the maximize the utilization of storage space.

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 server-side. 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.

Identifying files: 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. As shown in Figure 1, 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 (x_1, x_2, \dots, x_n) to probe deduplication respectively. If only file F_n does not upload the file chunk, the attacker will be very sure that the secret information is x_n . Because the server-side identifies the existence of the unique chunk x_n , 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 existence status can be used as the medium of communication.



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.

2 Background and Related Work

2.1 Deduplication in Cloud Storages

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 f_m , then the file f_m 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 f_n to own cloud folder. After deduplication check, he or she finds 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.

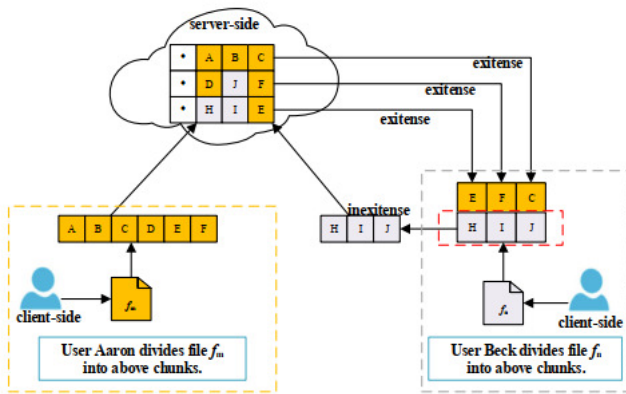


Figure 2: An example of deduplication check

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 storage space 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 client-side 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

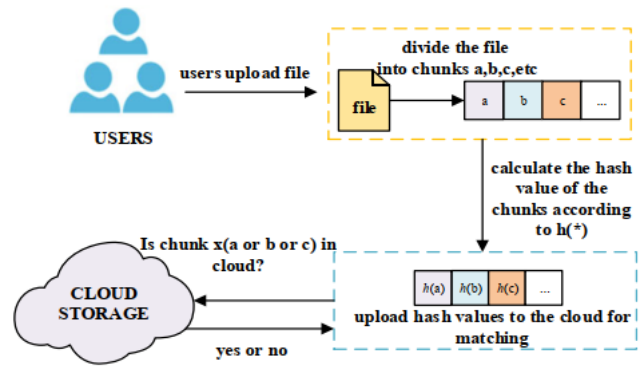


Figure 3: Deterministic response of data deduplication

channel, they recommend using a random threshold solution (RT). The server-side assigns a threshold t_x which is belongs to $[2, d]$. Thus, it is only known by the server-side. When a file chunk c_x is uploaded and the number of existing copies is greater or equal to t_x , 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 t_x , 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 t_i 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 scheme designs 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 used 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 server-side responses can be broken.

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, they are used to construct the DBTP in Table 2.

Table 2: List of notations used in the DBTP

Symbols	Description
f_x	The file that will to be uploaded
c_n	The chunks in which the file f_x is divided
φ	Chunk size
H	chunks that no longer need to be checked
K	K is a collection of all file chunks
tag	assists in deduplication checking
$h(x)$	Hash function

3.1 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 $[x_i, x_{i+1}]$ or $[x_i, \text{tag}]$. 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 1 means the opposite. As you know from the Figure 4, t_i denotes the number of chunks that the cloud requires client-side to upload. The t_1 represents both chunks c_1 and c_2 are not in the cloud storage, the t_2 represents c_1 is not but c_2 is in the cloud storage, and so on.

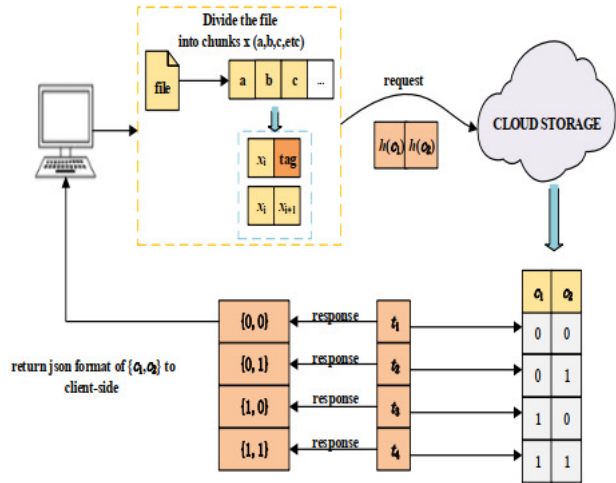


Figure 4: The process of the scheme

When the uploaded file is split, there are only two types of combinations. One is the combination of ordinary chunks like $[x_1, x_2]$, neither x_1 nor x_2 is a tag. The other is the combination of ordinary and tag chunks like $[x_i, \text{tag}]$. 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 t_4 , the others cannot equal 0. Otherwise, the chunk will be missing. Thus, DBTP needs to meet $t_1 \neq 0$, $t_2 \neq 0$, and $t_3 \neq 0$. Specifically, if the attacker is interested in chunk c_2 , he or she can upload two chunks of $[c_1, n]$ first. Thus, c_1 must exist in the cloud. Then, the attacker can upload $[c_1, c_2]$. Obviously, if $t_3 \neq t_4$ and the state value of c_1 both are 1, the existence state of c_2 can be easily determined. In this sense, $t_3 = t_4$ needs to be satisfied. Similarly, it can be obtained that $t_2 = t_4$ must also be satisfied.

It is easy to draw the following conclusions that $t_2 = t_3 = t_4$ and the values must be selected from 1 and 2. Thus, the value of $[t_1, t_2, t_3, t_4]$ 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 exposed the privacy of inexistence which c_1 and c_2 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 in Table 3.

Table 3: Design for DBTP

t	$[c_i, c_j]$	the chunk is uploaded by client-side
$t_1 = 1$	$[0,0]$	c_i or c_j
$t_2 = 1$	$[0,1]$	c_i
$t_3 = 1$	$[1,0]$	c_j
$t_4 = 1$	$[1,1]$	c_i or c_j

3.3 DBTP Algorithm Description

The algorithm shows the details of DBTP. First, in order 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 information, 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, $h(c_i)$ and $h(c_j)$, 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.

4 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 E5-26xx

Algorithm 1 DBTP

```

1: Begin
2: client partitions  $f_x$  into chunks  $c_1, \dots, c_n$ .
3: create a chunk tag =  $c_{n+1}$ .
4: while  $i, j \in K$  and  $i, j \notin H$  do
5:   random selection of two chunks  $[c_i, c_j]$  and  $i \neq j$ .
6:   if  $i == j$  then
7:      $h(c_j) = h(\text{tag})$ .
8:   end if
9:   client performs deduplication on  $[h(c_i), h(c_j)]$ .
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
23:    client uploads  $c_i$  or  $c_j$  for Table 2.
24:     $H = H \cup c_i$  or  $c_j$ 
25:  end if
26:  if client receives  $[c_i, c_j] = [0,1]$  then
27:    client uploads  $c_i$  for Table 2.
28:     $H = H \cup c_i$  and  $c_j$ 
29:  end if
30:  if client receives  $[c_i, c_j] = [1,0]$  then
31:    client uploads  $c_j$  for Table 2.
32:     $H = H \cup c_i$  and  $c_j$ 
33:  end if
34:  if client receives  $[c_i, c_j] = [1,1]$  then
35:    client uploads  $c_i$  or  $c_j$  for Table 2.
36:     $H = H \cup c_i$  and  $c_j$ 
37:  end if
38: end while
39: End

```

v4. DBTP is implemented by Python 3.7.6 platform and MySQL database.

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 in Figures 5, 6 and 7.

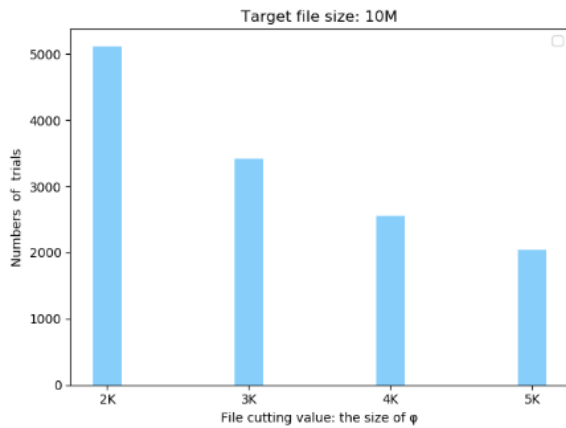


Figure 5: Target file size(10M)

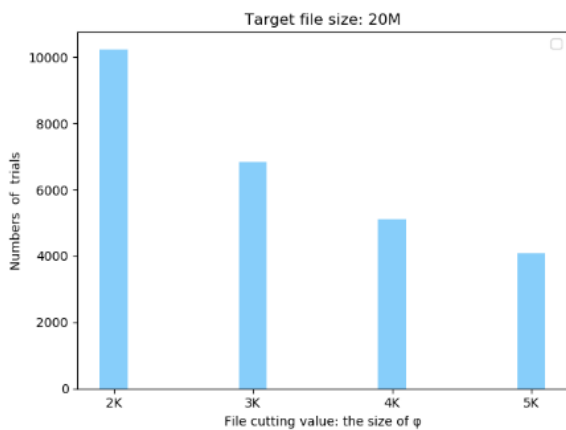


Figure 6: Target file size(20M)

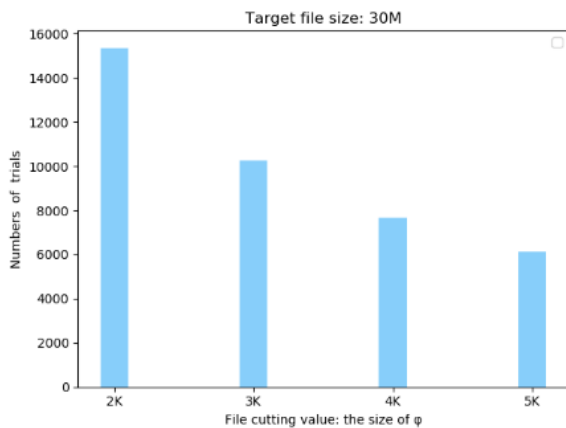


Figure 7: Target file size(30M)

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 that was 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 the deduplication strategy, but also by chunk size.

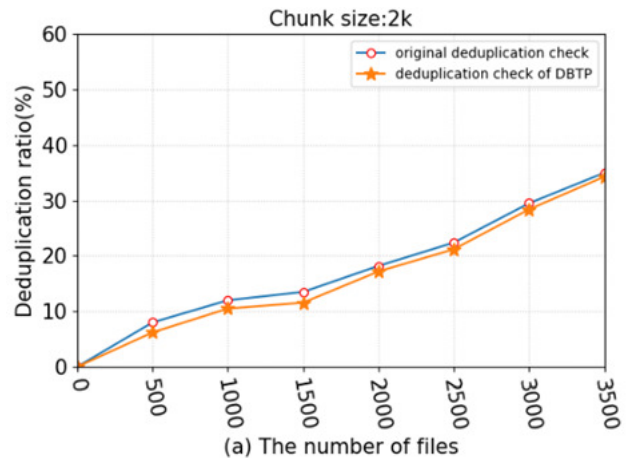


Figure 8: Deduplication ratio comparison(2k)

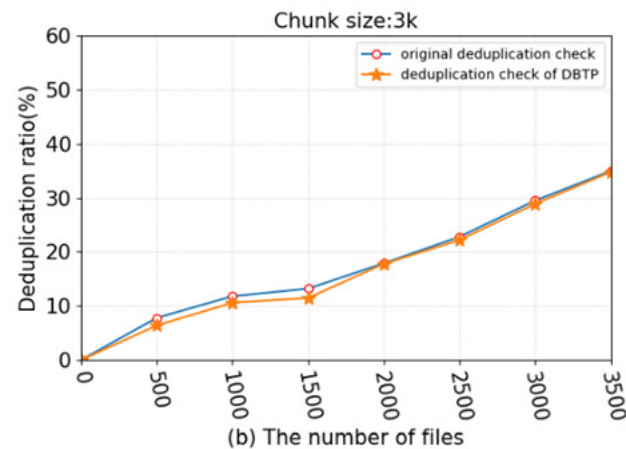


Figure 9: Deduplication ratio comparison(3k)

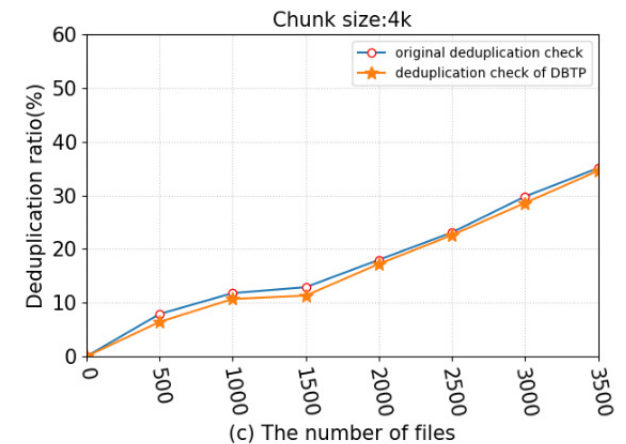


Figure 10: Deduplication ratio comparison(4k)

4.3 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 comparison experiment between DBTP and the original schema deduplication rate. Based on Figures 5, 6, and 7, we examine 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 transmission 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 deduplication ratios of the two methods. It is obvious that the original deduplication check has relatively higher deduplication ratio compared to the DBTP method. Because the original deduplication check finds the maximum opportunity to eliminate the redundant chunk. Fortunately, deduplication check of DBTP guarantees the two-side privacy of data, but the deduplication ratio is only a little reduced, and the gap between them is within an acceptable range. In general, DBTP precisely guarantees a good balance between deduplication efficiency and data privacy protection.

5 Conclusions

Although cloud storage service providers have widely adopted cross-user client-side deduplication check to reduce 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 paper, we propose a solution, DBTP, based on tag chunk for auxiliary transmission. This scheme leaks zero-knowledge for side channel. In other words, it can prevent the attacker from gaining the existence or inexistence status information from deduplication check. DBTP implements a stronger two-side privacy and performance guarantee based on minimal modification of the ordinary deduplication mechanism.

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, Communication and Computing Technology*, pp. 35–43, 2017.
- [2] Z. Guo, "Cryptanalysis of a certificateless conditional privacy-preserving authentication scheme for wireless body area networks," *International Journal of Electronics and Information 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," *IEEE Security & 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 storage," in *International Conference on ICT Convergence (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*, vol. 46, pp. 250–258, 2019.
- [8] J. Liu, N. Asokan, and B. Pinkas, "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.
- [9] D. Meister and A. Brinkmann, "Multi-level comparison of data deduplication in a backup scenario," in *Proceedings of SYSTOR: The Israeli Experimental Systems Conference*, pp. 8, 2009.
- [10] D. T. Meyer and W. J. Bolosky, "A study of practical deduplication," *ACM Transactions on Storage (TOS'12)*, vol. 7, no. 4, pp. 14, 2012.
- [11] 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.
- [12] Z. Pooranian, K. C. Chen, C. Mu Yu, and M. Conti, "Rare: Defeating side channels based on data-deduplication in cloud storage," in *IEEE INFOCOM IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS'18)*, pp. 444–449, 2018.
- [13] P. Puzio, R. Molva, M. Önen, and S. Loureiro, "Cloudedup: Secure deduplication with encrypted data for cloud storage," in *IEEE 5th International Conference on Cloud Computing Technology and Science*, vol. 1, pp. 363–370, 2013.

- [14] S. Shan, "An efficient certificateless signcryption scheme without random oracles," *International Journal of Electronics and Information Engineering*, vol. 11, no. 1, pp. 9–15, 2019.
- [15] Y. Shin and K. Kim, "Differentially private client-side data deduplication protocol for cloud storage services," *Security and Communication Networks*, vol. 8, no. 12, pp. 2114–2123, 2015.
- [16] V. Turner, J. F. Gantz, D. Reinsel, and S. Minton, "The digital universe of opportunities: Rich data and the increasing value of the internet of things," *IDC Analyze the Future*, 2014. (<https://www.emc.com/leadership/digital-universe/2014iview/index.htm>)
- [17] D. Wang, D. He, P. Wang, and C. H. Chu, "Anonymous two-factor authentication in distributed systems: Certain goals are beyond attainment," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 4, pp. 428–442, 2014.
- [18] D. Wang and P. Wang, "Two birds with one stone: Two-factor authentication with security beyond conventional bound," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 708–722, 2016.
- [19] D. Wang, N. Wang, P. Wang, and S. Qing, "Preserving privacy for free: Efficient and provably secure two-factor authentication scheme with user anonymity," *Information Sciences*, vol. 321, pp. 162–178, 2015.
- [20] W. Xia, H. Jiang, D. Feng, F. Douglis, P. Shilane, Y. Hua, M. Fu, Y. Zhang, and Y. Zhou, "A comprehensive study of the past, present, and future of data deduplication," *Proceedings of the IEEE*, vol. 104, no. 9, pp. 1681–1710, 2016.
- [21] W. Xia, H. Jiang, D. Feng, and Y. Hua, "Silo: A similarity-locality based near-exact deduplication scheme with low ram overhead and high throughput," in *USENIX Annual Technical Conference*, pp. 26–30, 2011.
- [22] W. Xia, Y. Zhou, H. Jiang, D. Feng, Y. Hua, Y. Hu, Q. Liu, and Y. Zhang, "Fastcdc: A fast and efficient content-defined chunking approach for data deduplication," in *{USENIX} Annual Technical Conference ({USENIX}{ATC} 16)*, pp. 101–114, 2016.
- [23] C. M. Yu, S. P. Gochhayat, M. Conti, and C. S. Lu, "Privacy aware data deduplication for side channel in cloud storage," *IEEE Transactions on Cloud Computing*, 2018. (<https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8260900>)

Biography

Jie Ouyang received the bachelor degree in engineering from Nanchang HangKong University, Nanchang, Jiangxi Province, China in 2014. Now, he is a master student in software engineering at the School of Computer Engineering and Science, Shanghai University. His research interests include Network Security, Cloud computing platform, Data mining and analysis. Email:oyjcoding@shu.edu.cn.

Huiran Zhang received the B.S. degree and the Ph.D. degree from the University of Toyama, Japan. He is currently an Associate Professor with the Shanghai University, Shanghai, China. His research interests include Cloud Computing, Machine Learning, and big data semantic processing.

Hongqing Hu has been learning at Shanghai University for her master's degree from September 2017 to the present. She is focusing on the research of cloud computing and machine learning.

Xiao Wei received the B.S. degree from the Shandong University, China, and the Ph.D. degree from Shanghai University, China, all in computer science. He is currently an Associate Professor with the Shanghai University, Shanghai, China. His research interests include NLP, machine learning, textual semantic analysis, and big data semantic processing.

Dongbo Dai is a lecturer at Shanghai University. His main research interests are computational theory and algorithmic game theory.