Secure Single-Server Fuzzy Deduplication without Interactive Proof-of-Ownership in Cloud

Shuai Cheng, Shengke Zeng*, Yawen Feng, Jixiang Xiao
School of Computer and Software Engineering
Xihua University
Chengdu, China
scheng98@163.com

Haoyu Zheng
Xihua College
Xihua University
Chengdu, China
RicardoZhy@outlook.com

***Abstract*—The redundant of multimedia data made an unnecessary waste in encrypted cloud storage, unlike text with completely consistent content, multimedia data allows a certain degree of similarity in deduplication, In this work, we focus on the multimedia data which takes a seriously proportion of storage in scenarios such as data outsourcing to propose secure fuzzy deduplication without the additional servers based on Convergent Encryption(CE), say the Single-server Fuzzy Deduplication (SSFD). Compared to the related fuzzy deduplication, SSFD is strong at resisting brute-force attacks caused by server-server collusion, moreover, we also put server-client collusion attacks into security solutions. Additionally, to enhance the security of data, the proposed scheme provides both protection against replay attacks and verification of label consistency and adds no extra communication such as Proof of Ownership(PoW) in interaction. We separately presented a formal security analysis and performed performance at last to prove security solutions and evaluate the experimental results, it shows SSFD provides both a reliable fuzzy images secure deduplication protocol and a computationally feasible solution.**

***Index Terms*—Cloud storage, fuzzy deduplication, single-server, proof of ownership, convergent encryption.**

I. Introduction

Cloud servers greatly facilitate data owners to save local storage, At the same time, there is a considerable proportion of redundant data in these outsourced servers, and redundancy in backup and archival storage systems is significantly more than 90% [1]. Among these redundant data, nearly-identical files occupied an eye-catching portion. The purpose of secure fuzzy deduplication is to delete the large quantity of identical or almost identical files or blocks [2] [3] [4] (including videos, pictures, audio, etc.) uploaded, and only securely keep one copy on the server [5], in short, the deduplication technology can both avoid unnecessary consumption of computation and storage burden [6] [7] while providing a well-protected on data. Deduplication technology has become a common means of saving communication overhead and storage capacity for most cloud service providers(including Amazon S3, Bitcasa and Microsoft Azure [8]) today. This way of retaining financial resourcing for cloud service provider shows data secure deduplication great significance [9].

The first scenario of deduplication came up in 2000 [10], however, it is difficult to compare ciphertexts encrypted by users with different private keys. To solve this problem, Convergent Encryption (CE) [11] extracts the secret key from the plaintext for encryption and decryption. Bellare *et al.'s* [12] proposed Message-Locked Encryption (MLE), which calculates the encryption key by combining the unencrypted data and system parameters. Time has come to today, there is a variety of deduplication mechanisms have been proposed [8] [13] [14] [15] [16], however, these plans do not apply to fuzzy data effectively, the difference between similar files and completely identical files in the above deduplication scheme reflects on they only allow the identical-same documents in deduplication.

Thus, to further reduce data redundancy in storage, Li *et al.'s* [4] and Chen. *et al.'s* [17] both proposed a secure deduplication scheme for similar images. However, their schemes are based on sharing a group key and can only remove redundant images within this group. For a system that interacts with multi-user and cross-domain, the scope of this deduplication mechanism is heavily limited. Jiang.*et al.'s* [18] proposed a FuzzyMLE and a FuzzyPow scheme, which can effectively solve the data deduplication, where FuzzyMLE has security enhancements ith an aided server, conceiving a situation that collusion may exist between the storage server and aided server, the server-aided-based scheme is not reliable mechanisms against brute force attacks. For seeking profit, the aided server and the storage server definitely can leak data to each, giving exposure to stored information which belongs to clients. Additionally, as Jiang.*et al.'s* mentioned about the process of FuzzyPow, once the server obtain both ciphertext c_w and the similar one c'_w, it will be able to compute plaintext w as $w \oplus w' = c_w \oplus c'_w$ with a similar file w'. Although Oblivious Transfer(OT) is used to protect c_w, actually, it still takes a risk in operating leak-prohibited ciphertext.

Liu *et al.'s* [14] raised a scheme that only relies on one single server to deduplicate for the first time. However, it takes no consideration of collusion between the server and clients, and it's a pity that this whole deduplication scheme only works for exactly the same files rather than similar data. A new single-server deduplication scheme was proposed by Takeshita *et al.'s* to try to solve the problem of nearly-identical Image deduplication [2]. As Takeshita *et al.'s* argued in the article, the scheme of them cannot resist brute force attacks carried by repeatedly querying both server and clients, moreover, the experimental evaluation of Takeshita *et al.'s* only show the time cost rather than the effect of deduplication in quantity so we cannot give a frank judgment to admire this deduplication mechanism is efficient. There are huge pits in single-server fuzzy deduplication still not filled.

To overcome the above-mentioned challenges, in this paper, we implemented a secure single-server fuzzy deduplication mechanism without extra interactive proof-of-ownership. It mainly consists of three contributions as following:

- We conceive a strategy that uses a single server, compared to the existing single-server scheme, we aim at ending the brute force attack caused by the sever cooperation with the uploader, since the privacy of the client summoned by the uploader may be revealed under this collaboration.
- We pose a proposal that only allows the clients who hold the original value of $phash$ being able to recover the key of the encrypted similar images. No need for extra interactively challenges and responses like PoW, and we set an additional verification mechanism in server to maintain the consistency of the tag.
- The security analysis and experimental evaluation of SSFD are shown at the end. The precision of our deduplication rate is well-performing derived from experimental results. Besides, SSFD is competent to be against online brute-force attacks by restricting times of communication. It can provide thorough privacy protection while completing the outstanding data deduplication which is based on a large amount of experimental data.

In subsequent articles, system models and design goals will be discussed in Section 2, then, we will describe preliminaries in Section 3, after that, we have an exhaustive introduction to the scheme and system in Section 4, and security analysis and performance evaluations are respectively depicted in Section 5 and Section 6, finally, the conclusion will be concluded in Section 7.

II. Models

A. System Model

Different from most existing cloud (cloud computing) related research schemes [2] [19] [20], our system relies on no additional servers. The whole entity of SSFD consists of three entities: an uploading client, who outsources its image files to the cloud server, a cloud service provider who is responsible for storing private pictures and need long-term preservation in this interactive protocol, and a number of summoned client $C_i = \{C_1, C_2, ..., C_n\}$, where n is the number of clients summoned by the uploading client, and C_i is the first uploader of the file, they will be gathered together for aiding the uploading client to obtain the symmetry key. Figure 1 illustrates the system architecture.

B. Threat Model

An honest-but-curious server S will comply with the process of deduplication faithfully while it provides honest

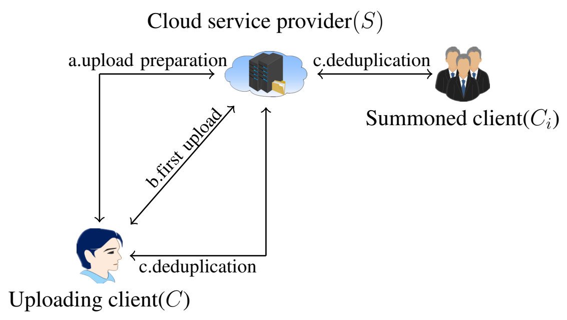


Fig. 1. System model

storage services, on the other hand, it is curious about the plaintext of encrypted data and attempts to learn the plaintext by Internal off-line guess attacks.

A malicious uploading client C attempts to defraud the server and possess the ownership of files that do not belong to itself by transmitting data that has been used or expired data. In addition, C can guess the content of the plaintext uploaded by C_i in the way of exchanging the parameters and other information provided by C_i through federating with S.

A malicious summoned client C_i may provide incorrect information during the stage of comparison, resulting in redundant or even loss of data in the S. Besides, it can also deceive C and have guessing attacks by the means of colluding with S.

III. Preliminaries

A. Hamming Distance and Threshold

Haming Distance. Hamming distance represents the number of two characters in positions corresponding to two (of the same length) strings.

Definition 1 (Hamming Distance). *For x and y over a finite binary $\mathcal{F}^n$, where $x = \{0,1\}^n, y = \{0,1\}^n$ the Hamming distance denoted d(x, y), is defined as the number of positions at which x and y differ, i.e., $d(x,y) = \sum_1^n x_i \oplus y_i$, where x_i, y_i respectively from ith position of x and y.*

Threshold. Hamming distance is commonly used to indicate the degree of similarity between two strings.

Definition 2 (Threshold). *We set D to represent the threshold of the Hamming distance to judge similarity or not as below:*

1) *If $d(x,y) = 0$, x and y are identified as identical strings*
2) *If $d(x,y) \leq D$, x and y are identified as similar strings.*
3) *If $d(x,y) > D$, x and y are identified as dissimilar strings.*

B. Perceptual Hashing

Perceptual hashing($phash$) is a one-way mapping that transforms multimedia data sets into perceptual summary sets, it detects images with the same perception content [21].

Definition 3 (Perceptual Hashing). *Here are the properties of $phash$:*

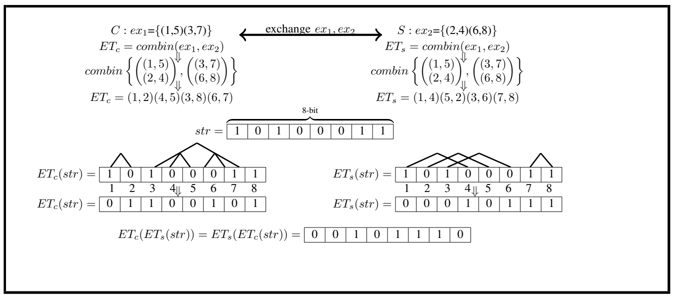
1) *Digest similarity: For x and x' are two images operating on finite collection of plaintext space $\mathcal{K}$, the value extract by $phash$ denoted $phash(x), phash(y)$, there have $d(phash(x), phash(x')) \leq D$ when x is similar to x', else will be $d(phash(x), phash(x')) > D$ on $\mathcal{K}$.*

2) *One-way: it can calculate $phash(x)$ with a given x, but the process is irreversible that is computationally unfeasible to find the input x when given the output $phash(x)$.*

We resize the image into a fixed-size gray matrix, and then partition the image matrix after DCT changes into blocks, next, calculate the entropy value of each block. Finally, we connect them into a string in as the value of the perceptual hash. Due to the limited pages of this conference, again, we do not provide a detailed description and relevant parameters about this process, those who are interested can pay attention to our full version.

C. Exchange Rules

The exchange table is the main item among the exchange rules, it can protect the confidentiality of multiple strings during the comparison process. Figure 2 provides an example of convert which used an 8-bit binary sequence under the exchange rules.



$C : ex_1$={(1,5)(3,7)}
exchange ex_1, ex_2
$S : ex_2$={(2,4)(6,8)}
$ET_c = combin(ex_1, ex_2)$
$ET_s = combin(ex_1, ex_2)$
$combin\left\{\begin{pmatrix}(1,5)\\(2,4)\end{pmatrix}, \begin{pmatrix}(3,7)\\(6,8)\end{pmatrix}\right\}$
$ET_c = (1,2)(4,5)(3,8)(6,7)$
$ET_s = (1,4)(5,2)(3,6)(7,8)$
8-bit
$str =$ 1 0 1 0 0 0 1 1
$ET_c(str) =$ 1 0 1 0 0 0 1 1
1 2 3 4 5 6 7 8
$ET_c(str) =$ 0 1 1 0 0 1 0 1
$ET_s(str) =$ 1 0 1 0 0 0 1 1
1 2 3 4 5 6 7 8
$ET_s(str) =$ 0 0 0 1 0 1 1 1
$ET_c(ET_s(str)) = ET_s(ET_c(str)) =$ 0 0 1 0 1 1 1 0

Fig. 2. An example of convert which used an 8-bit binary sequence under the exchange rules

Definition 4 (Exchange Rules). *C and S are the participants in deduplication, they each provide a set of position pairs, which we call the exchange factor(ex) here. For the purpose of avoiding repeat positions, the positions provided by one party must be even-numbered, so the other party can only provide odd-numbered positions. As the two parties receive each other's ex, they rearranged the position to generate an exchange table(ET) by using the exchange function $combine()$ which randomly combines the received ex_1 and their local ex_2 follow the corresponding ordinal position. Finally, c and s will get their own rearranged ET respectively called ET_c and ET_s. We will use $ET(str)$ to define an operation hat exchanges string str with table ET, and ET has the following properties.*

- *Comparability: for a string str and two exchange tables ET_c and ET_s generated by the same set of exchange factors, there is*
$$ET_c(ET_s(str)) = ET_s(ET_c(str)).$$
- *Reversibility: for a string str and exchange tables ET_c, there is*
$$ET_c(ET_c(str)) = str.$$

IV. Proposed Scheme

A. Upload Preparation

Clients who want to upload files in this system must experience upload preparation as follow steps to know the target clients(who may have similar files) and whether their files need further deduplication.

Step1: C and S generate their private exchange table. C and S will select half of the positions as exchange factors respectively according to the length of the perceptual hash(e.g., 256 bit).

- C and S form their exchange factor ex_c and ex_s separately by randomly combining odd positions and even positions, then they exchange the ex_c and ex_s to each
- When they receive the exchange factor from another side, C and S respectively compute
$$ET_s = Combin(ex_c, ex_s), ET_c = Combin(ex_c, ex_s),$$
where $Combin(\cdot)$ is the random combination operation for exchange factors.

Through the above-mentioned interactions, C and S formed their unique and private exchange table ET_c and ET_s.

Step2: C and S double exchange phash. In order to achieve the confidentiality of the information and enable effective comparison, we process the extracted value of $phash$ as follows.

- C and S respectively extract the perceptual hash value ph_c, ph_s from the image I and I_s by the following formula:
$$ph_c = phash(I),\ ph_s = phash(I_s).$$
where I the uploading image of C, I_s the unified and unique compared image of S, and $phash(\cdot)$ is the perceptual hash extraction operation.
- Then they make the first exchange of ph_c and ph_s with their own private exchange table ET_c and ET_s to obtain $ET_c(ph_c)$ and $ET_s(ph_s)$, next, S sends $ET_s(ph_s)$ the to C.
- When C receives $ET_s(ph_s)$, it computes Eph_s which is exchanged twice with the following formula and sends $Eph_s, ET_c(ph_c)$ to S.
$$Eph_s = ET_c(ET_s(ph_s)).$$
- Finally, S obtains both Eph_s which be double exchanged, and $ET_c(ph_c)$ which is exchanged once, then it exchanges the latter one again by the following formula:
$$Eph_c = ET_s(ET_c(ph_c)).$$

The exchanged $phash$ swapped by the exchange table can be safely transmitted in the channel, moreover, even if an original binary sequence and its exchanged state are known to the adversary, the probability of guessing the exchange table can be negligible.

Step3: S performs a deduplication check. for the purpose of confirming whether the uploading image is a duplicate file and determining whether to perform a further deduplication operation, S performs the following operations.

- S calculates the Hamming distance d_c of two double exchanged perceptual hashes Eph_c, Eph_s with the following formula.
$$d_c = d(Eph_c, Eph_s).$$
where $d(\cdot)$ is the Hamming distance calculation operation.
- If there have all d_{ci} make $|d_c - d_{ci}| > D$, where D is the threshold in deduplication and d_{ci} is the Hamming distance stored in the S and compared I_c with the previously uploaded image, then they will enter the first upload phase.
- If there exists any d_{ci} that makes $|d_c - d_{ci}| \leq D$, then they will enter the deduplication phase.

B. First Upload

If there are no duplicate files in the cloud, then S will ask C to upload image I after the deduplication check. The specific process is as follows.

- S instructs C to upload image I with the message First Upload, which means that S does not store a file similar or identical to I.
- when C received the instruction, it will encrypt uploading Image I by using $phash$ of I as the encryption key as below:
- Then C will send ID_c of itself and C_I to S, after S obtains the information, it will record $\{d_c, C_I, ID_c, ET_s, ET_c(ph_c), Eph_s, ex_c, ex_s\}$ as a information link with C .
$$C_I = SE_{ph_c}(I).$$
where $SE_k(\cdot)$ is the symmetric encryption algorithm with a key k.

C. Deduplication

When C enters the deduplication phase, it means that the S may have stored an image similar to the uploading one, so C needs to interact with C_i to figure out whether files are duplicated in the help with S.

Step1: C interacts with C_i with a new exchange table. After S narrowed down the scope of target clients which may have the same or identical image, we need a more precise comparison between C and C_i. So we need a new round of exchanges as follows.

- According to the Hamming distance d_{ci} which is within the threshold D, S will find the record $\{C_{Ii}, ID_{ci}, ET_{si}, ET_{ci}(ph_{ci}), Eph_{si}, ex_{ci}, ex_{si}\}$ which are linked to C_i.
- Instead of exchanging with ET_{si}, S computes the exchanged $phash$ value of its ph_s by using ET_{ci} which belongs to uploaded Client C_i as
$$ET_{ci}(ph_s) = ET_{si}(Eph_{si}),$$
where Eph_{si} is the double exchanged $phash$ of C_i.
- Then S sends $\{ID_{ci}, ET_{ci}(ph_{ci}), ex_{ci}, ex_{si}, ET_{ci}(ph_s)\}$ to C, C generates a new exchange table ET'_c by computing
$$ET'_c = Combin(ex_{ci}, ex_{si}).$$
- Based on this new exchange table ET'_c, C swaps ph_c of its image and obtains $ET'_c(ph_c)$, then it has to double exchange ph_{ci} which is swapped by ET_c for the first time, and ph_s which are also swapped by ET_c. the double exchange formula is
$$Eph'_{ci} = ET'_c(ET_{ci}(ph_{ci})), Eph'_{si} = ET'_c(ET_{ci}(ph_s)),$$
- C sends $\{Eph'_{ci}, Eph'_{si}\}$ to S, then S keeps Eph_{ci} and Eph_{si} which are exchanged twice. After that, C delivers $ET'_c(ph_c)$ to C_i according to ID_{ci}.
- C_i finished the double exchange of ph_c by using its own private table ET_{ci} as
$$Eph'_c = ET_{ci}(ET'_c(ph_c)).$$

Step2: S validates the exchange table of C_i. So far, C and C_i have co-generated Eph'_c, Eph'_{ci} and Eph'_{si} by using a new double exchange for ph_c, ph_{ci} and ph_s, Eph'_c and Eph'_{si} will be compared in this step to judge whether C_i is exchanged with the correct exchange table ET_{ci}. The specific process is as follows.

- C_i sends the $phash$ Eph'_c which has double exchanged to S.
- When S received both Eph'_c and Eph'_{si}, it will calculate the Hamming distance d'_c between them as
$$d'_c = d(Eph'_{si}, Eph'_c).$$
- If d'_c equals d_c, where d_c is the Hamming distance between ph_c and ph_s calculated in the upload preparation phase, then S will assume that C_i used the correct exchange table ET_{ci} and judge it as an honest client.
- If d'_c not equals d_c, then S will assume that C_i used the incorrect exchange table and judge it as a dishonest client, then the remaining operations will be terminated.

Step3: S starts a further deduplication check. After confirming that C_i provided the correct exchange table, S will check the Hamming distance between the $phash$ value of the uploaded image and the compared one to form a final determination as to whether the file is duplicated. Procedures can be described as follows.

- S will calculate the Hamming distance d'_{ci} between the two double exchanged $phash$ Eph'_c and Eph'_{ci} by computing
$$d'_{ci} = d(Eph'_c, Eph'_{ci}).$$
- If there are all $d'_{ci} > D$, then S will decide that it did not store a duplicate file of image I , and the first upload phase with the C.
- If there is a $d'_{ci} \leq D$, then S will identify I as a duplicate file of a stored image belonging to C_i, then it will compute The XOR value of the double exchanged $phash$ which we also named the position difference value P as
$$P = Eph'_c \oplus Eph'_{ci}$$

- Then S sends url_I to client C, where url_I is the public resource locator of C_I. Meantime, S sends P to C_i.

Step4: C_i verifies P and inverses exchange P for once. C_i needs to prevent C and S from obtaining the private $phash\ ph_{ci}$ with using unrelated images by verifying the similarity among these images. After this verification with the right result, C_i will have an inverse exchange on P with its own table ET_{ci}. The operation will be performed according to the following procedure.

- To sum the number d_p of 1 in P, which also is the Hamming distance between ph_c and ph_{ci}, C_i calculates
$$d_P = \sum_{i=1}^{256} P_i$$
where P used here is 256-bit binary sequence and $P = \{P_1, P_2, ..., P_{256}\}$.
- If $d_P > D$, then C_i will determine that there is abnormal behavior in the system and quit the process of deduplication.
- If $d_P \leq D$, then C_i will perform an inverse exchange of P using its own table ET_{ci} as
$$ET'_c(ph_c \oplus ph_{ci}) = ET_{ci}(P).$$
- In the end of this step, C_i delivers $ET'_c(ph_c \oplus ph_{ci}$ directly to C.

Step5: C recovers and verifies the symmetric key. In this step, C needs to recover the symmetric key ph_{ci} by using the received data and ph_c. In addition, to prevent C_i and S from conspiring to deceive C, C also needs to verify the authenticity of the ph_{ci}. The process can be followed.

- C will find the corresponding ciphertext C_I which is stored on the server according to this url_I and keep it temporarily local.
- After being given the $ET'_c(ph_c \oplus ph_{ci})$, C recovers the key as
$$ph_{ci} = ET'_c(ET'_c(ph_c \oplus ph_{ci}) \oplus ph_c.$$
- ph_{ci} will be used as the symmetric key to decrypt C_i and C verifies ph_{ci} by computing
$$phash(De_{ph_{ci}}(C_i)),$$
$$d(ph_{ci}, ph_c).$$
- If both $ph_{ci} == phash(De_{ph_{ci}}(C_i))$ and $d(ph_{ci}, ph_c) \leq D$, it would be a smooth and reasonable situation which mean C got the correct decryption key, then it will preserve only $\{ph_{ci}, url_i\}$ and drop others to save storage.
- Otherwise, C will determine that there is abnormal behavior in the system and quit the process of deduplication.

Through the above verification, C can effectively prevent C_i and S from conspiring to use an unrelated the image to obtain the information about image owned by C, and avoid further guessing attacks.

V. Security Analysis

In this section, we give a careful analysis of the security properties of our scheme. We infer that the data confidentiality is guaranteed by proving the security of the exchange table ET, then we show that the probability of acquiring the ET is negligible to prove the collusion resistance, and we describe the infeasibility of online and offline guesses respectively to show that resistance to brute-force attacks. Finally, resistance to replay attacks and tag consistency are proven by analyzing security of communication in the deduplication system and feasibility of the tag verification mechanism separately. Unfortunately, due to space constraints, we do not describe them in detail here. Those who are interested can pay attention to our full version.

VI. Performance Evaluations

A. Simulation Results

Feature extraction time with phash. We selected an image with 180KB and 1200 × 1005 pixels by using CC_WEB_VIDEO [22], then resize it to multiple sizes while maintaining the aspect ratio, and test the time spent in extracting features under the changed size for 50 times each. The result is shown in the Figure 3, we can see the time cost is less than 100ms at the image size of 4038 × 3417, moreover, in the normal size of 480 × 402 pixels, the time spending is closing 20ms which is an acceptable cost.

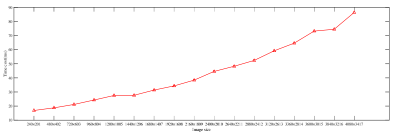


Fig. 3. Time cost of phash in various sizes

Variation of deduplication rate at several thresholds. In this simulation, we modify a certain proportion of pixels in random positions of the picture to random values. In addition, we also set various thresholds D (including D=30, D=35, D=40), and the phash length is set to 256 bits. We know from Figure 4, observing that under a certain degree of modification, these thresholds have a suitable deduplication rate, where D=40 takes the best effect which has the deduplication rate remains above 0.9 When the percentage of modified pixels is lower than 2%.

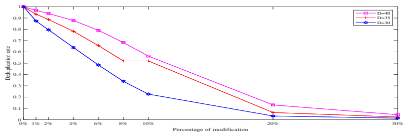


Fig. 4. Deduplication rate $\mathcal{P}_{dr}$ affected by modifying pixels in various thresholds

Influence of various distortions on the deduplication. Considering the distortion occurs frequently In the transmission, we tested in this simulation under the influence of circular mean filter blur, motion filter blur, Gaussian noise, and salt&pepper noise with 20505 images, as the experimental results of $\mathcal{P}_{dr}$ with exact data have shown in Figure 5, where $\mathcal{P}_{dr}$ is the deduplication rate, and the calculation formula of the deduplication rate is given as $\mathcal{P}_{dr} = 1 - N/T$, we found that with the increase of the number of pictures, the deduplication rate $\mathcal{P}_{dr}$ still remained within a stable limit, which testify our system have a good robustness in face of conventional interference, so that the distortion changes caused by the above blurring and noise can be accurately detected within a certain range, and the threshold we chose for this experiment is D=40.

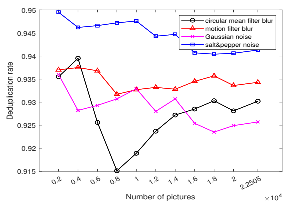


Fig. 5. Deduplication rate $\mathcal{P}_{dr}$ influenced by various distortion

Comparing to the Existing Fuzzy Deduplication Scheme [18] We compared the deduplication rate of this scheme with the existing scheme in this part with a different threshold(D=30,D=40,D=50), which we both using the non-replicated images from the same dataset of CC_WEB_VIDEO with similar numbers respectively are 22317 of Jiang.*et al.'s* [18] and 22505 of our proposal. Notably, due to the distinction in the algorithms, the selected thresholds of the corresponding algorithms selected should also be different, where the compared thresholds selected by Jiang are t=1, t=2 and t=3. We set the pixel modification value of the picture to 1.5%, and the detail of this comparison is shown in the Figure 6.

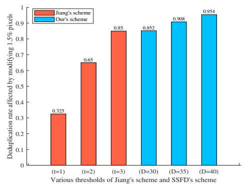


Fig. 6. Deduplication rate $\mathcal{P}_{dr}$ affected by modifying 1.5% pixels in various thresholds

The experimental result shows that despite the modification of 1.50%, our deduplication rate remains above 85%. Compared with the Jiang‘s scheme which set t=3, although the deduplication rates are seen almost the same, however, we slightly higher under this influence.

VII. Conclusion

This paper designs a single-server fuzzy deduplication scheme mainly aims at images which does not require redundant file ownership authentication interaction. In order to securely deduplicate files, we design a novel $phash$ function and exchange table ET to extract the image features and be as the blinding tags respectively. Both security analysis and experimental evaluation show that our scheme can resist collusion or other forms of brute force attacks, and can effectively verify the consistency of labels and prevent repeat attacks to ensure data confidentiality. Additionally, SSFD also has a satisfactory deduplication effectiveness with its deduplication rate.

References

[1] H. Biggar, “Exeperiencing data de-duplication: improving efficiency and reducing capacity requirements,” 2007.

[2] J. Takeshita, R. Karl, and T. Jung, “Secure single-server nearly-identical image deduplication,” 2020.

[3] M. Chen, S. Wang, and L. Tian, “A high-precision duplicate image deduplication approach.,” *J. Comput.*, vol. 8, no. 11, pp. 2768–2775, 2013.

[4] X. Li, J. Li, and F. Huang, “A secure cloud storage system supporting privacy-preserving fuzzy deduplication,” *Soft Computing*, vol. 20, no. 4, pp. 1437–1448, 2016.

[5] Y. Tian, S. M. Khan, D. A. Jiménez, and G. H. Loh, “Last-level cache deduplication,” in *Proceedings of the 28th ACM international conference on Supercomputing*, pp. 53–62, 2014.

[6] H. Hovhannisyan, K. Lu, R. Yang, Q. Wen, and W. Mi, “A novel deduplication-based covert channel in cloud storage service,” in *GLOBECOM 2015 - 2015 IEEE Global Communications Conference*, 2014.

[7] N. Mandagere, P. Zhou, M. A. Smith, and S. Uttamchandani, “Demystifying data deduplication,” in *Proceedings of the ACM/IFIP/USENIX Middleware'08 Conference Companion*, pp. 12–17, 2008.

[8] R. Di Pietro and A. Sorniotti, “Proof of ownership for deduplication systems: a secure, scalable, and efficient solution,” *Computer Communications*, vol. 82, pp. 71–82, 2016.

[9] 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.

[10] 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.

[11] J. R. Douceur, A. Adya, W. J. Bolosky, P. Simon, and M. Theimer, “Reclaiming space from duplicate files in a serverless distributed file system,” in *Proceedings 22nd international conference on distributed computing systems*, pp. 617–624, IEEE, 2002.

[12] M. Bellare, S. Keelveedhi, and T. Ristenpart, “Message-locked encryption and secure deduplication,” in *Annual international conference on the theory and applications of cryptographic techniques*, pp. 296–312, Springer, 2013.

[13] S. Keelveedhi, M. Bellare, and T. Ristenpart, “{DupLESS}:{Server-Aided} encryption for deduplicated storage,” in *22nd USENIX security symposium (USENIX security 13)*, pp. 179–194, 2013.

[14] 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.

[15] Y. Shin, D. Koo, J. Yun, and J. Hur, “Decentralized server-aided encryption for secure deduplication in cloud storage,” *IEEE Transactions on Services Computing*, vol. 13, no. 6, pp. 1021–1033, 2017.

[16] Y. Zhang, C. Xu, N. Cheng, and X. Shen, “Secure password-protected encryption key for deduplicated cloud storage systems,” *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 4, pp. 2789–2806, 2021.

[17] L. Chen, F. Xiang, and Z. Sun, “Image deduplication based on hashing and clustering in cloud storage,” *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 15, no. 4, pp. 1448–1463, 2021.

[18] T. Jiang, X. Yuan, Y. Chen, K. Cheng, L. Wang, X. Chen, and J. Ma, “Fuzzydedup: Secure fuzzy deduplication for cloud storage,” *IEEE Transactions on Dependable and Secure Computing*, 2022.

[19] Z. Yan, L. Zhang, D. Wenxiu, and Q. Zheng, “Heterogeneous data storage management with deduplication in cloud computing,” *IEEE Transactions on Big Data*, vol. 5, no. 3, pp. 393–407, 2017.

[20] X. Liu, K.-K. R. Choo, R. H. Deng, R. Lu, and J. Weng, “Efficient and privacy-preserving outsourced calculation of rational numbers,” *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 1, pp. 27–39, 2016.

[21] X. Yang, R. Lu, J. Shao, X. Tang, and A. Ghorbani, “Achieving efficient secure deduplication with user-defined access control in cloud,” *IEEE Transactions on Dependable and Secure Computing*, 2020.

[22] Z. Yan, W. Ding, X. Yu, H. Zhu, and R. H. Deng, “Deduplication on encrypted big data in cloud,” *IEEE transactions on big data*, vol. 2, no. 2, pp. 138–150, 2016.