2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th Intl. Conference on Data Science and Systems

PTS-Dep:A High-Performance Two-party Secure Deduplication for Cloud Storage

Wenlong Tian [1,2], Ruixuan Li [1], Weijun Xiao [2], and Zhiyong Xu [3,4]

[1]School of Computer Science and Technology, Huazhong University of Science and Technology, China
[2]Electrical and Computer Engineering, Virginia Commonwealth University, USA
[3]Math and Computer Science Department, Suffolk University, USA
[4]Shenzhen Institute of Advanced Technology, Chinese Academy of Science, China

***Abstract*—In cloud storage, the message-locked encryption method is widely used in security deduplication. However, Brute force attack becomes a serious issue. Current research addresses the brute force attack problem in secure deduplication using a third-party model. Even though there is a trusted third party in real life, it is hard to be applied to traditional two-party based deduplication system which only includes the client and the storage provider. It is obvious that industries prefer to take the simpler and more practical secure architecture under the same level of security. However, the existing two-party secure deduplication approaches either have inferior performance or security holes. To make the two-party secure deduplication comparable in performance with unprotected baseline and keep the same level security with the existing two-party secure deduplication, we propose a high-performance two-party secure deduplication, PTS-Dep. By leveraging secure duplicate data detection scheme and secure duplicate data's key sharing scheme, PTS-Dep can perform data deduplication with the security guarantee. Our approach improves average deduplication performance up to 92% for Fslhome workloads compared to previous secure deduplication schemes when the average chunk size is 12KB.**

***Keywords*-Secure Deduplication; Two-Party; Secure Privacy Sharing; Secure Duplicate Data Detection**

I. Introduction

As the popularity of cloud storage increases, more and more people prefer to outsource personal data through cloud storage service for flexibility and reliability. There are a large number of redundant data across users. Cloud storage providers take traditional deduplication methods to remove redundancy to utilize storage space efficiently. However, performing data dedupliaction on plaintext will have information leakage. Even though a user can encrypt their data by their own key to protect privacy, it sacrifices the deduplication capability of cloud storage because the same plaintext may have different ciphertext.

In order to close the gap between security and data deduplication, Bellare el at. propose the message-locked encryption(MLE) Scheme [1]. The key, in MLE scheme, used for encryption and decryption is derived from the content, which allows data deduplication on ciphertext space [2], [3], [4], [5], [6], [7], [8], [9]. Obviously, the same content data, even uploaded by different user, has an equivalent encryption key, and vice versa. However, message-locked encryption is easily attacked by brute force attacks, since it is keyless that the ciphertext's encryption key is derived from plaintext content. Once attackers acquire a low entropy template of target ciphertext, it can guess plaintext from the template. Then, attackers can generate an encryption key for each guessing based on MLE Scheme. After encryption for each guessing plaintext by the corresponding generated encryption key, attackers can compare their cipertext with the target ciphertext. Once found an equal one, the target's plaintext is hacked.

To resist brute force attacks, the server-aid scheme is proposed in [10] to encrypt under message-based keys obtained from a trusted key-server via oblivious PRF protocol. It can achieve a high security as long as the key server remains inaccessible to the attacker. However, When the key server is compromised with others, the security in Dupless[10] will be degraded to message-locked encryption that cannot resist brute force attacks. Some other previous efforts are also trying to solve this problem [11], [12], [13], [14], [15], [16], [17]. These research work can generally be categorized into two types of architectures, third-party and two-party models. The third-party model includes clients, a storage provider, and a third trusted key server. The second one only contains the clients and server.

Actually, no third party can be trusted as Bruce Schneier has pointed out, after the 2013 mass surveillance disclosures [18]. What's more, the existing well-known cloud storage is doing deduplication based on two-party model only including the client and server. Storage providers would like to choose an easily deployment without changing the existing architecture while considering the same level of security. However, the existing work in two-party secure deduplication is not feasible due to low performance or security holes. Liu el at. proposes a scheme to directly do deduplication between clients and servers by combining PAKE and homomorphic encryption method, but the performance will suffer a lot as the number of users increasing [13]. Yu el at. proposes an encryption deduplication scheme by combining Merkle puzzle as a negotiation method between clients and servers, but there are security holes [16]. Since the duplicate data detection is done by the client and server together, a malicious client can utilize the detection process in [16]

978-1-5386-6614-2/18/$31.00 ©2018 IEEE
DOI 10.1109/HPCC/SmartCity/DSS.2018.00122

to substitute the ciphertext of a popular file as a malicious script.

In summary, it is meaningful by considering the practical deployment in designing the secure deduplication protocol. But, there is a contradiction between practical implementation and security. It is because some complex security computation designs greatly degrade the system performance, which is not practical in real scenarios. In this paper, we propose a high-performance two-party secure deduplication, PTS-Dep. The performance of PTS-Dep outperform 92% in overall performance compared with the latest two-party secure deduplication mechanism with the same level of security. Our main contributions in this paper are summarized as follows:

- Firstly, we identify two main challenges in two-party secure deduplication by analyzing the existing approaches. The first challenge is related to practical secure redundant data detection. The second challenge is associated with securely and practically sharing redundant data's key with other data owners.
- Secondly, we propose a practical secure duplicate data detection scheme. A special data structure in this scheme can quickly detect redundant data in a secure way. Moreover, attackers can difficultly do off-line brute force attacks based on the detection data structure itself.
- Thirdly, we also propose a practical secure duplicate data's key sharing scheme among data owners who do not need to be always online. Compared with [13], this method can significantly improve the performance of two-party secure deduplication.
- Finally, we implement a high performance two-party secure deduplication scheme, PTS-Dep. We also conduct comprehensive experiments to evaluate the performance of PTS-Dep. By comparing with the unprotected baseline [1] and the state-of-the-art two-party secure deduplication system [13], experimental results show that our method improves the average performance up to 92% compared with the state-of-the-art two-party secure deduplication systems[13] and is close to the performance of [1].

The rest of the paper is organized as follows. In Section II, we analyze the problem of the latest secure deduplication work in the two-party scenario. Then, we propose a solution for secure deduplication in the two-party scenario in Section III. Finally, we present experimental results in Section IV. In Section V, we discuss the related work and conclude the paper in Section VI.

II. Problem Statement

It is much more difficult to design a two-party scheme without a trusted key server because there are more security issues that need to be addressed. For example, an encryption key sharing process can be applied to the trusted third party which may prevent user privacy leakage. However, it must consider an effective and secure protocol to reach the goal of practical secure key sharing in the two-party scenario. We summarize the two-party secure deduplication into following two issues.

The first issue is **how to practically do redundant data detection while also ensure the security?** The duplicate data detection is based on a hash function that is content-based, in [1]. It will be used as identification for brute force attack by attackers. To keep the security of redundant data detection, some random property should be added into redundant data detection. we called it as key-based. However, it has an effect on deduplication capability under two party scenario. Different clients will generate different random property for each block. Even though previous work [13] adds random property into duplicate data detection, the redundant data detection is done based on the online collaborative computation between multi-clients and server. It is obvious that requiring the data owners always online is not practical in reality.

The second issue is **how to securely share the duplicate data's encryption key with other corresponding data owners while also taking the system performance into consideration?** Since distinct clients have different encryption key for each chunk, data owner needs to share the corresponding encryption key with other data owners who also own the redundant chunk under two-party scenario. Otherwise, other data owners cannot decrypt the ciphtertext of redundant chunk. What's more, if a securely sharing process suffers the system a lot, it is not acceptable for practical implementation even though it can hold the security property. For example, in [13], Password Authenticated Key Exchange(PAKE) method is used to exchange the user's encryption key among the data owners. However, PAKE will lead to a lot of TCP traffic when performing block-level deduplication with large number of users. Even though it proposes a checker scheme to certainly decrease the overhead of PAKE, it can not fundamentally prevent performance degradation caused by PAKE.

A. Challenges in Designing Secure Deduplication

In this subsection, we summarize the challenges in designing secure deduplication. Then, we compare our scheme with existing secure deduplication efforts in Table I. It is noted that the threat model of PTS-Dep is the same as the security model in [13].

1) Independent Trusted Server Assumption(C1): Choosing two-party model or three-party secure deduplication model directly impacts the following detailed designing consideration. "—" denotes three-party model. "$\surd$" means two-party model.
2) Brute Force Resilience without Security Hole: Turning the keyless property to key-based property is the core idea to resist brute force attack. "—" denotes it exist

Table I
THE STATE-OF-THE-ART CLASSIFICATION

Methods	C1	C2	C3
CE[11] & MLE[1]	√	—	√
tCE[15]	—	√	—
DupLESS [10]	—	√	—
EwS[12]	—	√	—
ClouDedup [14]	—	√	√
SVCDedup[17]	—	√	—
PAKEDedup [13]	√	√	—
XDedup [16]	√	—	√
PTS-Dep	√	√	√

Table II
NOTATIONS

Notation	Description
k_g	Message-locked encryption key for chunk g
k_f	Message-locked encryption key for chunk f
E_f	the ciphertext of chunk f
r_{m1}	the first random number at m-th time
r_{m2}	the second random number at m-th time
r_{n1}	the first random number at n-th time
r_{n2}	the second random number at n-th time
$sh(k_f)$	the short hash of k_f
$Vitem_f$	the validation data structure for f
$Trans_f$	the data structure for transferring privacy with those who has the same chunk with f
S_Trans_f	part of $Trans_f$ which is stored at server
p	the 1024-bit prime number which is public
$Tag(E_f)$	the hash value of E_f
S	a collection that each $Vitem_i$ element has the same short hash, $sh(k_i)$
$\phi(n)$	number of all positive integer up to a given integer n that are relatively prime to n

some problems in Brute Force Resilience without Security Hole. "√" means it can securely handle brute force attack.

3) Interaction Complexity & Computation Difficulty(C3): Performance is an important issue as security in secure deduplication. Interaction complexity and computation difficulty is the main impact factors in performance. "—" denotes the complicated interaction & computation. "√" means simple interaction & computation.

III. PROPOSED SCHEME

A. Overview of PTS-Dep

The workflow of PTS-Dep is shown in Figure 1. To simplify the discussion, we denote entities in our scheme as Table II. Since the whole system performs the block-level deduplication, each file is split into chunks at the client side. Only the unique chunk will be stored at the server. Then, we study the following two major chunk operations: upload and restore.

When client i uploads a chunk f, the message-locked encryption key k_f and two random numbers , r_{m1} and r_{m2}, are generated where $k_f = Hash(f)$. In addition, data structures $Vitem_f$ and $Trans_f$ corresponding for f are produced locally based on k_f,r_{m1} ,r_{m2} and p where p is a public large prime number. The detailed calculation process is discussed in subsection III-B and III-C. Then, client i sends $Vitem_f$ and S_Trans_f, part of $Trans_f$, to the server. Next, the server filters out a small collection S based on the short hash section in $Vitem_f$. The short hash is generated by choosing the first fixed number of bits of k_f, which denotes as $sh(k_f)$. Each item x in S will have a corresponding $Vitem_x$ and S_Trans_x.

Therefore, the server securely determines whether f is a unique chunk or not by secure duplicate data detection scheme. Once f is a unique chunk, the server store $Vitem_f$ and S_Trans_f in the server. Then, client i generate a new key-based encryption key by Xor(k_f,$r_{m2}\%p$) where "Xor" denotes an exclusive or function and "%" denotes the modulation operation. The ciphertext of chunk f, E_f, is produced by symmetrically encryption based on the new key-based encryption key. After calculating the tag Tag(E_f), client updates the file metadata and uploads the E_f and Tag(E_f) to the server. If f is a duplicate chunk, the server sends the S_Trans_g and Tag(E_g) to client i, where chunk g is the same as f. Then, client i can acquire the key-based encryption key of g by secure key sharing method based on $Trans_f$ and S_Trans_g, which will be discussed in Subsection III-C.

When client i restores chunk f, the ciphertext of f is downloaded firstly based on the metadata. Then, the plaintext of f can be achieved by decryption with the corresponding encryption key. When dealing with restoring file, it can restore the file based on chunk sequence after achieving each chunk's plaintext. Since we do not focus on the key management issue in this paper, we assume that all keys for files and chunks are stored at the client side. It is noted that our proposed method is compatible with the randomize method proposed by [10] and [13].

B. Practical Secure Duplicate Data Detection Scheme

In this subsection, we present a practical secure duplicate data detection method by proposing a data structure $Vitem$. The $Vitem$ is combined based on the modular property[19], Euler's totient function[20] and discrete logarithm problem[21] together.

As shown in Figure 2, there are six parts in $Vitem_f$ for a chunk f. Note that, k_f is generated by SHA-256 hash with the chunk content. Two random numbers, r_{m1} and r_{m2}, are generated based on Guassian distribution and the range of values is ($2^{256}-1$,$+\infty$). The reason why keeps each random number larger than k_f is to guarantee the correctness of secure duplicate data validation method. The first part $sh(k_f)$ is used to reduce the search scope in duplicate data detection process because the redundant chunks have the same short hash of k_f. While the other parts is constructed based on Euler's totient function[20] and discrete logarithm problem[21].

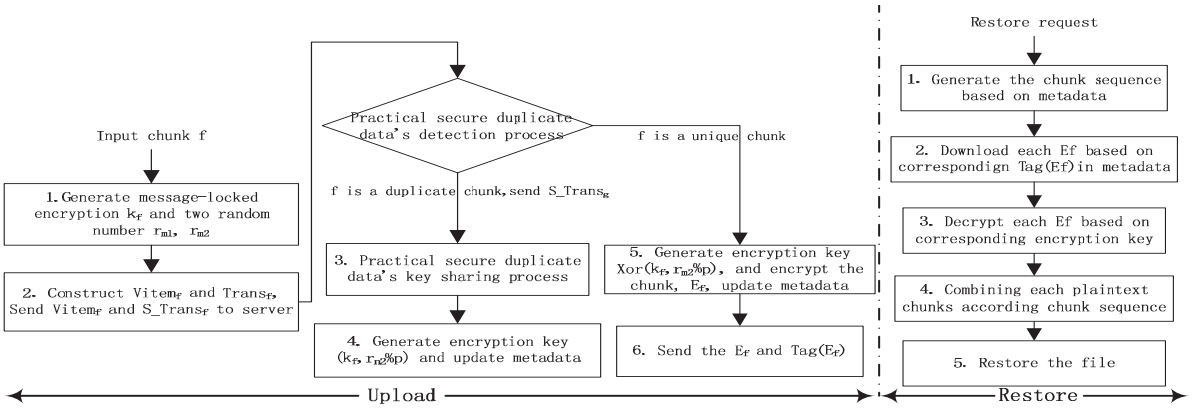


Figure 1. The Workflow of PTS-Dep

The first reason why we design $Vitem_f$ as Figure 2 is to avoid others doing brute force attack based on these data structure by adding the random property into each part of $Vitem_f$ except the $sh(k_f)$. For example, directly based on $sh(k_f)$ can not determine the k_f because of the remainder bits of k_f is unknown. Attackers can difficultly figure out random information from each part in $Vitem_f$, or by solving the solutions of constructing equations based on $Vitem_f$. Noted that, when p is 1024-bit or more, the discrete logarithm problem is difficult to be solved by conventional computer. $\phi(p)$ equals to $p-1$ since p is a large prime number.

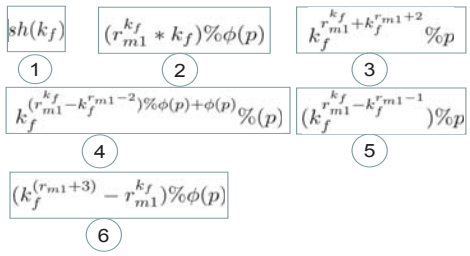


Figure 2. Vitem Data Structure

The second reason is that server can quickly determine whether f is a unique chunk or not based on our design. For simplicity, we assume there is a unique chunk g stored at server which is uploaded by client j. Then, client i uploads a chunk f to the server. As shown in Figure 3, the $Vitem$ data structure of chunk g and chunk j are (f_2,f_4,f_6,f_8,f_{10}) and (f_1,f_3,f_5,f_7,f_9), respectively. When client i uploads the $Vitem_f$ to server, $Vitem_g$ will be in the collection S since $sh(k_g)$ is equal to $sh(k_f)$. Then, the server calculates ($result1$,$result2$) based on Formulas 1 and 2. If f is a duplicate chunk for g, ($result1$,$result2$) must be equal to (1,1). If not, ($result1$,$result2$) should not be equal (1,1).

$$result1 = (((((f_4{}^{f_1+\phi(p)}\%p * f_5{}^{-f_2+\phi(p)}\%p)\%p * f_8{}^{f_9+\phi(p)}\%p)\%p * f_6{}^{f_1+\phi(p)}\%p)\%p * f_3{}^{-f_2+\phi(p)}\%p)\%p * f_7{}^{-f_{10}+\phi(p)}\%p)\%p \quad (1)$$

$$result2 = (((((f_3{}^{f_2+\phi(p)}\%p * f_6{}^{-f_1+\phi(p)}\%p)\%p * f_7{}^{f_{10}+\phi(p)}\%p)\%p * f_5{}^{f_2+\phi(p)}\%p)\%p * f_4{}^{-f_1+\phi(p)}\%p)\%p * f_8{}^{-f_9+\phi(p)}\%p)\%p \quad (2)$$

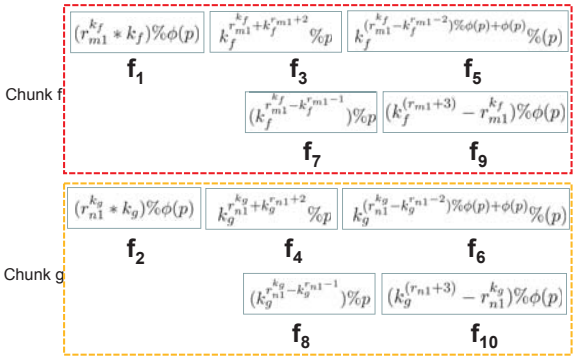


Figure 3. Vitem Example for Chunk f and Chunk g

C. Practical Secure Duplicate Data's Key Sharing Scheme

In this part, we propose a practical secure duplicate data's key sharing scheme. For each unique chunk, data owner uploads their S_Trans data structure into server. Once other clients uploads a duplicate chunk, the key-based encryption key of the duplicate chunk can be quickly shared among data owners. However, server can not figure out the random information in key-based encryption key by S_Trans data structure. What's more, there is no need to be online state for data owners compared with [13] during this sharing process.

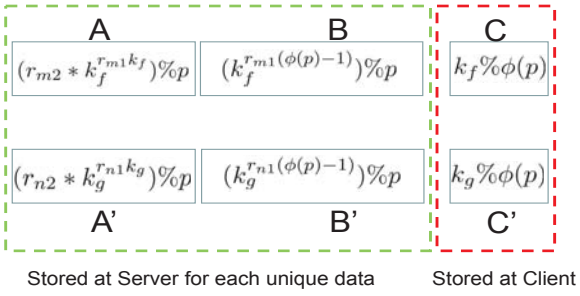


Figure 4. Trans' Data Structure

$$(A * B^{C'+\phi(p)})\%p = r_{m2}\%p \quad (3)$$

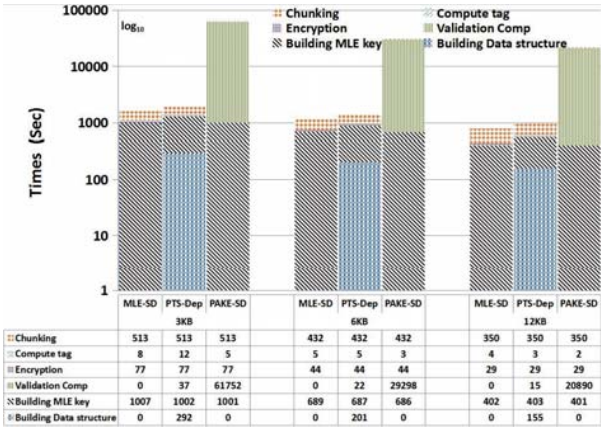
Specifically, we assume that client i uploads chunk g while the chunk f, uploaded by other client, is already stored at server. And g is the replicate chunk for f. As shown in Figure 4, the data structure of key sharing scheme has three parts. We denote that $Trans_g$ is (A',B',C') and S_Trans_g is (A',B') while $Trans_f$ is (A,B,C) and S_Trans_f is (A,B). It is note that server can only learn the S_Trans part since only S_Trans part is stored at server. When client i is notified by server that f is a redundant chunk for g based on practical secure duplicate data detection scheme, S_Trans_f is sent to client i. Note that, k_g is equal to k_f in $Trans_f$ and $Trans_g$ at this time. Based on Formula 3, client i can calculate the value of $r_{m2}\%p$. Then, the key-based encryption key can be derived from Xor(k_f,$r_{m2}\%p$).

IV. Performance Evaluation

A. Methodology

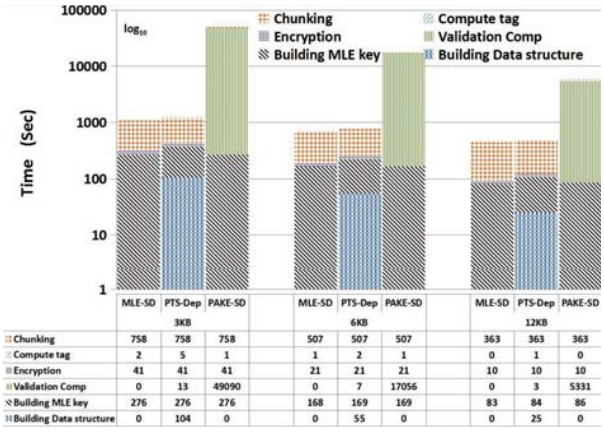
Experimental Setup All the experiments are conducted on a distributed platform. Each client is equipped with an Intel(R) Core(TM) i7-4790 @3.60GHz 8 core CPU, 16GB RAM, a 500GB 5400 rpm hard disk and is connected to a 100Mbps network. To compare proposed PTS-Dep with other schemes, we perform experiments on four datasets, including Linux Kernel [22], Fslhome [23], the video datasets [24] and Audio datasets [25]. We evaluate the performance of PTS-Dep compared with PAKE-based two-party secure deduplication [13](PAKE-SD) and the MLE two-party secure deduplicate [1](MLE-SD). It is because PAKE-SD is the state-of-the-art two-party secure deduplication system. And MLE secure deduplication system is treated as the unprotected baseline of two-party secure deduplication.

Metrics we identified the following metrics to evaluate proposed scheme. 1) Building Data Structure: Different systems have various cost for constructing data structures. This metric is to measure time consumption in constructing data structures. Note that, the data structure in PTS-Dep includes $Vitem$ and $Trans$. 2) Building MLE key: it



	MLE-SD	PTS-Dep	PAKE-SD	MLE-SD	PTS-Dep	PAKE-SD	MLE-SD	PTS-Dep	PAKE-SD
	3KB			6KB			12KB		
Chunking	513	513	513	432	432	432	350	350	350
Compute tag	8	12	5	5	5	3	4	3	2
Encryption	77	77	77	44	44	44	29	29	29
Validation Comp	0	37	61752	0	22	29298	0	15	20890
Building MLE key	1007	1002	1001	689	687	686	402	403	401
Building Data structure	0	292	0	0	201	0	0	155	0

Figure 5. The Performance of Linux Workloads



	MLE-SD	PTS-Dep	PAKE-SD	MLE-SD	PTS-Dep	PAKE-SD	MLE-SD	PTS-Dep	PAKE-SD
	3KB			6KB			12KB		
Chunking	758	758	758	507	507	507	363	363	363
Compute tag	2	5	1	1	2	1	0	1	0
Encryption	41	41	41	21	21	21	10	10	10
Validation Comp	0	13	49090	0	7	17056	0	3	5331
Building MLE key	276	276	276	168	169	169	83	84	86
Building Data structure	0	104	0	0	55	0	0	25	0

Figure 6. The Performance of Fslhome Workloads

denotes the time consumption of producing message-locked encryption keys. 3) Validation Comp: it means the time consumption for duplicate data's detection and duplicate data's key sharing. 4) Encryption: we use this metric to gather statistics of all unique chunks' encryption time by the key-based encryption. 5) Compute Tag: it denotes the time consumption by calculating the file and chunk's tag. 6) Chunking: the time consumption of chunking process in the system.

B. Numerical Results and Discussions

Firstly, we conduct the experiments in Linux workloads. As shown in Figure 5, the overall time consumption for each method decreases as the average chunk size changed from 3KB to 12KB. It is because the larger average chunk size it has, the less chunk number it generates. However, our scheme can have a close performance with the unprotected baseline, MLE-SD, which outperforms PAKE-SD under Linux workloads. For example, the overall time cost in

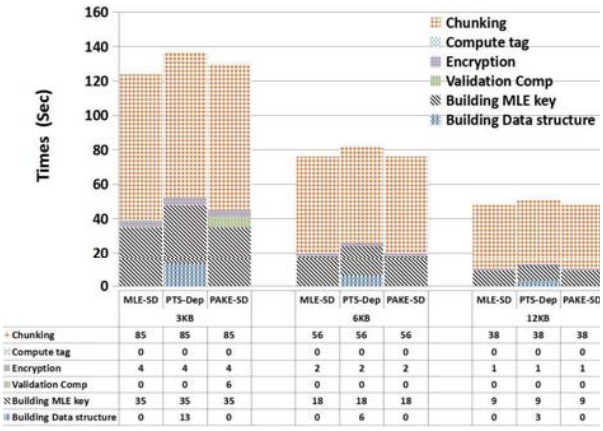
PAKE-SD costs is about 32 times as much as PTS-Dep when average chunk size is 3KB. Even though overall time consumption of each method is decreasing as the increase of average chunk size, the overall time consumption of PAKE-SD is still about 22 times as much as our scheme when the average chunk size is 12KB.

The reason is that too much Password Authenticated Key Exchange process occur in PAKE-SD even though it has a checker Selection scheme. Note that, PAKE procedure need two participants to do online interactive which need at least two TCP/IP handshaking. Once a corresponding short hash item is found in PAKE-SD, the uploader will initiate a PAKE procedure with other clients(at least one client). Furthermore, there are many duplicate chunks in Linux workloads. Hence, there will cost a lot in PAKE procedure. However, the performance of PTS-Dep is independent of the number of chunks, which is consistent with the MLE-SD's performance. Hence, Our scheme not only has a better performance than PAKE-SD, but also is very close with baseline in performance under Linux workloads.

Secondly, we also measure the performance of Fslhome workloads. As shown in Figure 6, similar to the Linux workloads, the time consumption of Validation Comp in PAKE-SD becomes the main cost in the whole overhead because of the password authenticated key exchange property that we mentioned before. Even though the overall time consumption of PAKE-SD decreases as the increase of average chunk size, the time cost of PAKE-SD is still too much. For example, the overall time consumption of PAKE-SD is 11 times as much as our scheme when average chunk is 12kB. More specifically, the time cost of Building Data Structure and Validation Comp parts in PTS-Dep is less than the corresponding part in PAKE-SD. As shown in Figure 6, the time consumption of building validation structure and validation in PTS-Dep only takes 6% compared with the unprotected baseline, MLE-SD. Consequently, Our scheme has the extraordinary performance compared with PAKE-SD under the Fslhome workloads.

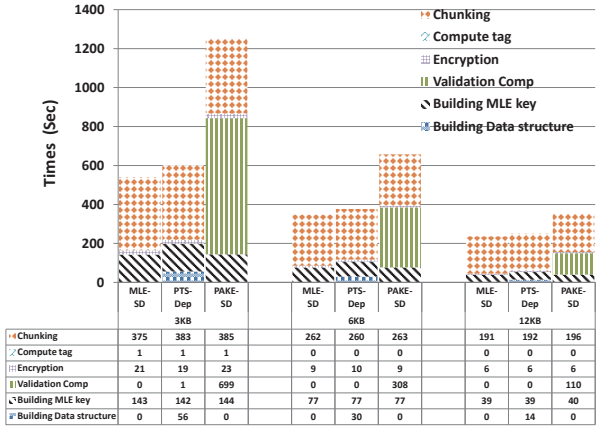
Thirdly, we also perform the experiments under the audio workloads and video workloads to simulate the performance under the low dedup ratio scenario. Figures 7 and 8 show the results under the audio workloads and video workloads, respectively. From Figure 7, we can find out that the performance of PAKE is almost the same with unprotected baseline, MLE-SD. The reason is that the audio workload has very low deduplication ratio. In other words, most of the uploading chunks are unique chunks. Hence, PAKE-SD will not bring too much PAKE procedures to sharing privacy among clients. Fortunately, the performance of our method is very close to the performance of baseline. Even the time cost of Building Data structure part is associated with the number of chunks, it is always less than the time consumption of Building MLE key part.

Finally, we measure the time consumption of Validation



	MLE-SD	PTS-Dep	PAKE-SD	MLE-SD	PTS-Dep	PAKE-SD	MLE-SD	PTS-Dep	PAKE-SD
	3KB			6KB			12KB		
Chunking	85	85	85	56	56	56	38	38	38
Compute tag	0	0	0	0	0	0	0	0	0
Encryption	4	4	4	2	2	2	1	1	1
Validation Comp	0	0	6	0	0	0	0	0	0
Building MLE key	35	35	35	18	18	18	9	9	9
Building Data structure	0	13	0	0	6	0	0	3	0

Figure 7. The Performance of Media(audio) Workloads



	MLE-SD	PTS-Dep	PAKE-SD	MLE-SD	PTS-Dep	PAKE-SD	MLE-SD	PTS-Dep	PAKE-SD
	3KB			6KB			12KB		
Chunking	375	383	385	262	260	263	191	192	196
Compute tag	1	1	1	0	0	0	0	0	0
Encryption	21	19	23	9	10	9	6	6	6
Validation Comp	0	1	699	0	0	308	0	0	110
Building MLE key	143	142	144	77	77	77	39	39	40
Building Data structure	0	56	0	0	30	0	0	14	0

Figure 8. The Performance of Uploading Media(video) Workloads

Comp when a new client repeatedly uploaded a chunk which is a duplicated chunk in cloud owned by various client number. As shown in Figure 9, we can clearly find out that the time consumption of Validation Comp part in PAKE-SD is increase sharply as the number of clients who owned the same duplicate chunk. The cost of validation process in PAKE-SD is still too expensive when the client number less than 8 compared with our scheme and MLE-SD. On the contrary, our-scheme is independent with the client number, which is the same with the unprotected baseline, MLE-SD. Consequently, our scheme greatly outperform PAKE-Scheme and has similar performance with the unprotected baseline, MLE-SD.

V. Related Work

In this section, we mainly discuss the related work about secure deduplication. We categorize these related work into two categories based on whether it has an independent server as a secure third-party.

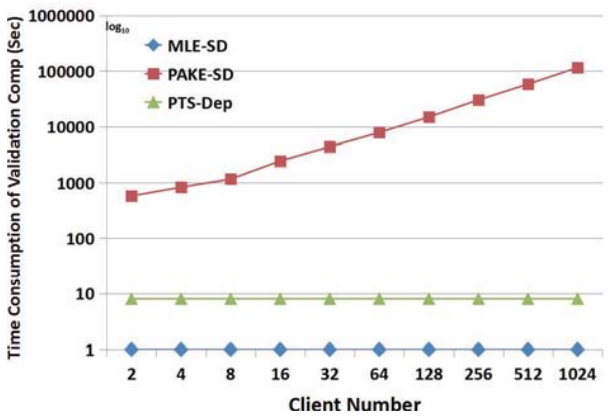


Figure 9. The Time Consumption of Validate Comp under Various Client Number

A. Secure Deduplication in Three-Party Model

In three-party secure deduplication, it assumes that exists a secure third party to be responsible for some secure process. To against the brute force attacks in secure deduplication, Bellare and Keelveedhi [10] present DupLESS to keep the key generation securely by introducing a secure third party server. More specifically, To guarantee that no one except client itself can derive the convergent encryption key as an identification during brute force attack, the secure third-party KS jointly compute a content-dependent key for every file by using oblivious pseudorandom function(OPRF). However, when putting the DupLESS into block-level secure deduplication, the OPRF scheme will dramatically degrade the whole performance in both clients and the third-party key server.

Based on the problem in DupLESS, SecDep [26] employs the User-Aware Convergent encryption and multi-level key management approaches to speed up the secure deduplication in third-party model. However, it can only do cross user's deduplication in file-level. But, when considering it into block-level deduplication, only insider user's deduplication can work. Even though it proposes some other contribution in key management, it still not solves the main problem in DupLESS. Similar to SecDep, a deduplication proxy sits in the middle of users and the cloud in [27], where the proxy and cloud perform cross-user deduplication but the user and proxy perform single-user deduplication. ClouDedup [14] works in a similar way. Threshold CE (tCE) [15] and PerfectDedup [28] perform the chunk-level deduplication by taking advantage of chunk popularity.

B. Secure Deduplication in Two-Party Model

Even though the secure third-party is very common in security scenario, the existing cloud storage provider is still not bringing the secure third-party into the deduplication scenario. In other words, there is only clients and storage provider in traditional deduplication scenario. Therefore, Storage provider rated considerations such as ease of deployment, when selecting a deduplication scheme. Consequently, some other work are dedicated to removing the third party in secure deduplication. Encrypt-with-Signature (EwS) [12] claims to eliminate the need for a key server and retain the security guarantee by using threshold signatures. Nonetheless, Zheng [17] argued that the dealers in EwS serve as the similar role of the key server in DupLESS.

Recently, Liu et al[13] propose PAKEDedup. It combines the Password Authenticated Key Exchange method and homomorphic encryption method to resist brute force attacks in the two-party scenario. However, it will degrade the secure deduplication performance since each duplication data's judgment at least cause a PAKE process. The situation becomes serious when considering it into block-level secure deduplication. Based on the PAKE performance problem, Xdedup[16] attempts to improve the performance of PAKE by introducing the symmetric encryption method into secure deduplication. However, there exist some security holes. since it move part of duplicate data detection process into client-side, malicious user can utilize this scheme substitute the other's uploaded data.

VI. Conclusions

In this paper, we summarize two critical issues by analyzing the problems in two-party secure deduplication. Then, three challenges are proposed around the practical implementation and the security. To securely and practically detect duplicate data and share the secret key in two-party secure deduplication scenario, we propose a high performance two-party secure deduplication, PTS-Dep. Furthermore, experimental results show that our scheme greatly improves the average deduplication performance up to 92% in Fslhome workloads. Additionally, PTS-Dep not only ensures the security, but also has a great improvement in performance compared with other previous two-party secure deduplication schemes.

VII. ACKNOWLEDGEMENT

This work is supported by the National Key Research and Development Program of China under grants 2016QY01W0202 and 2016YFB0800402, National Natural Science Foundation of China under grants 61572221, U1401258, 61433006 and 61502185, Major Projects of the National Social Science Foundation under grant 16ZDA092, Science and Technology Support Program of Hubei Province under grant 2015AAA013, Science and Technology Program of Guangdong Province under grant 2014B010111007 and Guangxi High level innovation Team in Higher Education InstitutionsInnovation Team of ASEAN Digital Cloud Big Data Security and Mining Technology. This work is also

partly supported by the National Science Foundation under grant CNS 1526190.

REFERENCES

[1] 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*. Springer, 2013, pp. 296–312.

[2] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer, "Farsite:federated, available, and reliable storage for an incompletely trusted environment," *Acm Sigops Operating Systems Review*, vol. 36, pp. 1–14, 2002.

[3] P. Anderson and L. Zhang, "Fast and secure laptop backups with encrypted de-duplication," in *International Conference on Large Installation System Administration*, 2010, pp. 1–8.

[4] J. Cooley, C. Taylor, A. Peacock, and F. Project, "Abs: the apportioned backup system," *Proceedings of the Csee*, vol. 31, no. 7, pp. 112–118, 2011.

[5] A. Zisman, "A static verification framework for secure peer-to-peer applications," in *International Conference on Internet and Web Applications and Services (ICIW 2007), May 13-19, 2007, Le Morne, Mauritius*, 2007, p. 8.

[6] L. Marques and C. J. Costa, "Secure deduplication on mobile devices," in *Proceedings of the 2011 workshop on open source and design of communication*. ACM, 2011, pp. 19–26.

[7] A. Rahumed, H. C. Chen, Y. Tang, P. P. Lee, and J. C. Lui, "A secure cloud backup system with assured deletion and version control," in *Parallel Processing Workshops (ICPPW), 2011 40th International Conference on*. IEEE, 2011, pp. 160–167.

[8] M. W. Storer, K. Greenan, D. D. Long, and E. L. Miller, "Secure data deduplication," in *Proceedings of the 4th ACM international workshop on Storage security and survivability*. ACM, 2008, pp. 1–10.

[9] Z. Wilcox-O'Hearn and B. Warner, "Tahoe: the least-authority filesystem," in *Proceedings of the 4th ACM international workshop on Storage security and survivability*. ACM, 2008, pp. 21–26.

[10] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Dupless: Server-aided encryption for deduplicated storage." *IACR Cryptology ePrint Archive*, vol. 2013, p. 429, 2013.

[11] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system," in *International Conference on Distributed Computing Systems*, 2002, pp. 617–624.

[12] Y. Duan, "Distributed key generation for encrypted deduplication: Achieving the strongest privacy," in *Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security*, 2014, pp. 57–68.

[13] 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*, 2015, pp. 874–885.

[14] P. Puzio, R. Molva, M. Onen, and S. Loureiro, "Cloudedup: Secure deduplication with encrypted data for cloud storage," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, vol. 1, 2013, pp. 363–370.

[15] J. Stanek, A. Sorniotti, E. Androulaki, and L. Kencl, "A secure data deduplication scheme for cloud storage," in *International Conference on Financial Cryptography and Data Security*. Springer, 2014, pp. 99–118.

[16] C.-M. Yu, "Poster: Efficient cross-user chunk-level client-side data deduplication with symmetrically encrypted two-party interactions," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1763–1765.

[17] Y. Zheng, X. Yuan, X. Wang, J. Jiang, C. Wang, and X. Gui, "Enabling encrypted cloud media center with secure deduplication," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, 2015, pp. 63–72.

[18] G. Greenwald and E. MacAskill, "Boundless informant: the nsas secret tool to track global surveillance data," 2013.

[19] E. Öztürk, B. Sunar, and E. Savas, "Low-power elliptic curve cryptography using scaled modular arithmetic," pp. 92–106, 2004.

[20] B. Kaliski, "Eulers totient function," pp. 430–430, 2011.

[21] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[22] Linux, "The linux kernel archives," https://www.kernel.org/.

[23] Fslhome, "Traces and snapshots public archive," http://tracer.filesystems.org/.

[24] C. Inc, "Coursera," https://www.coursera.org/.

[25] JAMENDO, "Jamendo music," https://www.jamendo.com/start.

[26] Y. Zhou, D. Feng, W. Xia, M. Fu, F. Huang, Y. Zhang, and C. Li, "Secdep: A user-aware efficient fine-grained secure deduplication scheme with multi-level key management," in *Mass Storage Systems and Technologies (MSST)*, 2015, pp. 1–14.

[27] P. Meye, P. Raipin, F. Tronel, and E. Anceaume, "A secure two-phase data deduplication scheme," in *High Performance Computing and Communications*, 2014, pp. 802–809.

[28] P. Puzio, R. Molva, M. Önen, and S. Loureiro, "Perfectdedup: Secure data deduplication," in *International Workshop on Data Privacy Management*. Springer, 2015, pp. 150–166.