PAPER

A Server-Based Distributed Storage Using Secret Sharing with AES-256 for Lightweight Safety Restoration

Sanghun CHOI†a), Shuichiro HARUTA†b), Yichen AN†c), *Student Members*, *and* Iwao SASASE†d), *Fellow*

SUMMARY Since the owner's data might be leaked from the centralized server storage, the distributed storage schemes with the server storage have been investigated. To ensure the owner's data in those schemes, they use Reed Solomon code. However, those schemes occur the burden of data capacity since the parity data are increased by how much the disconnected data can be restored. Moreover, the calculation time for the restoration will be higher since many parity data are needed to restore the disconnected data. In order to reduce the burden of data capacity and the calculation time, we proposed the server-based distributed storage using Secret Sharing with AES-256 for lightweight safety restoration. Although we use Secret Sharing, the owner's data will be safely kept in the distributed storage since all of the divided data are divided into two pieces with the AES-256 and stored in the peer storage and the server storage. Even though the server storage keeps the divided data, the server and the peer storages might know the pair of divided data via Secret Sharing, the owner's data are secure in the proposed scheme from the inner attack of Secret Sharing. Furthermore, the owner's data can be restored by a few parity data. The evaluations show that our proposed scheme is improved for lightweight, stability, and safety.

key words: *distributed storage, reed solomon, Secret Sharing*

1. Introduction

In these days, with the growth of the network and the portable devices, many people can easily access the Internet and enjoy a lot of Internet services. One of the most popular services is a WCS (Web Cloud Storage) [1] which is the centralized server storage such as Google Drive [2], Dropbox [3] and Sky Drive [4] as shown in Fig. 1(a). Thanks to the WCS and the Internet, whenever an owner can manage its data without the limitation of storage of its devices. Although the WCS provides convenience nowadays, it has the problems to concern for ensuring the safety of the owner's data [5]–[7]. Firstly as for the personal aspect, according to [8], [9], the personal data might be leaked by an adversary's cracking or a misbehaving of server storage's operator even if they are encrypted. This is because the personal data such as the photos are stored into the server storage. In addition, providing personal data to the investigation agency has already occurred. According to [10], Apple had already helped FBI agents to solve the criminal case by sharing stored relevant data in its cloud storage. Furthermore, according to [11], Google notes that the user's data can be provided to the government for investigating regulatory violations or criminal activity. Secondly, as for the company aspect, storing important data into single centralized storage is not recommended for security reason.

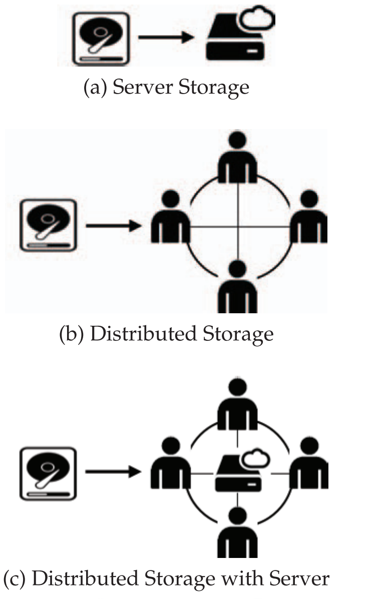


Fig. 1 Structure of the online storage services

To overcome the shortcomings mentioned above, for safely storing the owner's data in the storage, the distributed storage schemes have been studied [12]–[18], [20]–[22]. There are two types of distributed storage schemes. The first type is, the distributed storage schemes without the server storage as shown in Fig. 1 (b) which is called a P2P(Peer to Peer) storage [12]–[14]. The peers are the computer systems called the nodes which are connected to each other via the Internet [15]. The peer joins the P2P network as part of the distributed storage. The owner's data are separated with the encryption in the peer storages as the owner wants to store. Then, the divided data are sent to each of the peers which are the storage in the P2P as the number of the divided data. The owner's original data are restored with the decryption by gathering the separated data from the peer storages. However, since each peer storages must have the parity data for restoring the original data, the burden of data capacity on the peer will be grown by how much the parity data ensure

Manuscript received September 9, 2019.
Manuscript revised January 29, 2020.
Manuscript publicized April 20, 2020.

†The authors are with Dept. of Information and Computer Science, Keio University, Yokohama-shi, 223–8522 Japan.
a) E-mail: choi@sasase.ics.keio.ac.jp
b) E-mail: haruta@sasase.ics.keio.ac.jp
c) E-mail: anyichen@sasase.ics.keio.ac.jp
d) E-mail: sasase@ics.keio.ac.jp

DOI: 10.1587/transinf.2019EDP7243

the restoration. As for the second type, the distributed storage schemes with the server storage as shown in Fig. 1(c) have been investigated [16]–[18]. The parity data, to restore the owner's data from the leaking of peer storage, is stored via Reed Solomon code [19] into the server storage. So, the burden of data capacity on the peer storage is lesser than the first type. We assume that the distributed storage with the server storage scheme is suitable. However, in the distributed storage using Reed Solomon code, the counts and size of parity data are increased by how much the parity data are needed to restore the owner's original data from the leaking of the distributed storage. This occurs the total burden of limitation for the storage capacity. Moreover, the calculation time for restoration will be higher since many parity data are needed to restore disconnected data. To reduce the burden of data capacity and the calculation time for the distributed storage with the server storages, we have noted Secret Sharing storage schemes [20]–[22]. Shamir's Secret Sharing [23] has been studied to preserve the secret key in safe. The secret key is divided, encrypted and stored into the distributed storage. The secret key can be restored by using only a few pieces. By using these points, the owner's data are stored like the secret key in the distributed storage. However, there are two problems as follows: The burden of data capacity on the peer is high, the distributed data are easily restored if the pair of share on Secret Sharing is leaked.

We assume that if the problems mentioned above can be overcome, it will be suitable for the distributed storage with the server storages scheme. Therefore, we propose the server-based distributed storage using Secret Sharing with AES-256 for lightweight safety restoration. The divided owner's data will be safely kept by dividing on Secret Sharing with the server storage and AES-256 from leaking the pair of shares on Secret Sharing. The evaluations show that the burden of data capacity and the calculation time in the proposed scheme are lesser than the conventional distributed storage with server storage using Reed Solomon code and Secret Sharing. Furthermore, the restoration for when the divided owner's data are disconnected is more efficient. The rest of this paper is constructed as follows. The related work is explained in Sect. 2. The proposed scheme is described in Sect. 3. The results and evaluations are shown in Sect. 4. The discussion and limitation are described in Sect. 5. Finally, we conclude our research in Sect. 6.

2. Related Work

2.1 Distributed Storage without Server Storage

How to safely keep the owner's data in the distributed storage without the server storage, when the owner wants to store and restore its data, is important. One of the ways to safely keep the owner's data in the distributed storage is called P2P storage. In paper [12], the peer joins the P2P network as part of P2P storage. The owner's data are separated with the encryption by how much the owner wants to store its data into the peer storages. Then, the divided data are sent to each of the peer storage as the number of the divided data. The owner's data are restored with the decryption by gathering the separated data from the peer storages when the owner wants to restore its data. Each peer just controls P2P network flows between the owner and the peer storages. However, in that paper, how to restore the owner's data when the peer storages are disconnected is not considered. In papers [13], [14], to recover the disconnected data from the peer storages, the owner generates and stores the backup data for the peer storages. Therefore, each of the peer storages in the P2P helps each other to repair when one of the data in the P2P storage is broken. However, this occurs the burden of the data capacity to the peer storage since the peer storages have to take all of the backup data for other peers.

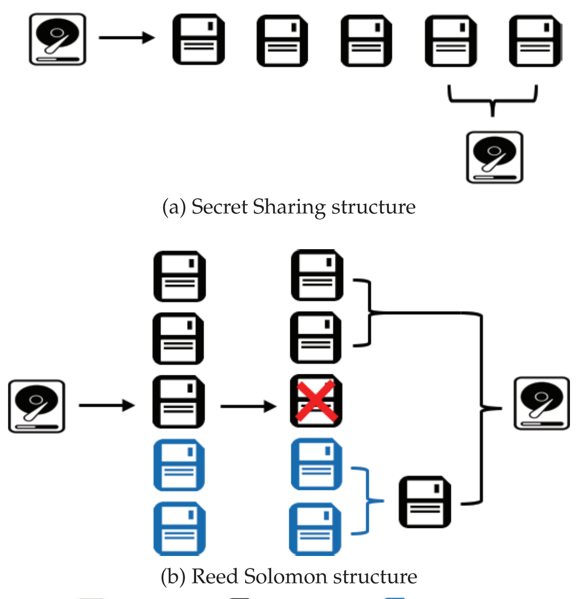


Fig. 2 Example of the conventional storages

To efficiently restore the owner's data from the leaking of the peer stroages, Secret Sharing [23] has been investigated for the distributed storage schemes [20]–[22]. There are two representative Secret Sharing types. Type 1 is called All-or-Nothing method [24]. The total size of share data is the same as the size of the original data in All-or-Nothing method. If the original data size is 10MB and divided into five parts, the total data size of five parts becomes 10MB. So, it looks that each peer storages have less data. If k, which is the value to restore the owner's data, is set 10, the owner's data will be divided into 10 shares in 10MB. In addition, all peer storages which have the divided data must always be connected to restore the owner's data. This means that if one of the shares is lost, the owner's data cannot be restored. That is the reason why All-or-Nothing method is not suitable for the distributed storage scheme. Type 2 is basic Secret Sharing as shown in Fig. 2(a). Most distribute storage schemes as Type 2 have been investigated [20]–[22]. The owner's data, which the owner wants to store in the distributed storage, are separated into parts data via Secret Sharing like the secret key is divided. The parts data include the parity data. The parts data are sent to each of the peer storages, not the server storage. In Fig. 2(a), if the owner's data are divided into five parts via Secret Sharing, if the parity data are set as two to restore the original data, the original data can be restored with only two parts via Secret Sharing. This means that the owner can restore its data with a few parity data. However, this occurs the increase of total data capacity since each of the size of separated data is the same as that of the original data via Secret Sharing. If the original data size is 10MB and divided into five parts, the total data size of five parts becomes 50MB. This is the serious burden of data capacity for the storage service. Moreover, the calculation time for restoring the owner's data by using the parity data is high. In addition, the attacker who knows the pair of shares on Secret Sharing can get the owner's data as the inner attack. The inner attack means that the leaking of the owner's data from peer storage in the distributed storage service via Secret Sharing. Those are the serious problems for the safety of the distributed storage.

2.2 Distributed Storage with Server Storage

In papers [16]–[18], Reed Solomon code [19] has been used for the distributed storage with the server storage to reduce the burden of data capacity for peer storages. Figure 2(b) shows the basic structure of Reed Solomon code. The HDD icon denotes the owner's original data. The black disk and blue disk denote the separated data and the parity data, respectively via Reed Solomon code. Before keeping the owner's data in the distributed storage, the owner divides its data into several pieces. Then, the owner generates the parity data for restoring its data when the divided data are disconnected. As shown in Fig. 2(b), if the parity parts are set as two via Reed Solomon code, one of the lost separated data can be restored by the parity parts via Reed Solomon code. By using Reed Solomon code for the distributed storage, when the data are disconnected from the peer storages, the disconnected data will be regenerated by the parity data. The parity is stored in the server storage to decrease the stored data size for the peer storages. This server storage is assumed to be alive always. The parity data are increased by how much the parity data can recover the disconnected data. This means that the total data size will be bigger as the distributed storage has more parity data to restore the many disconnected data. This is a burden of data capacity for the distributed storage. Moreover, all of the divided data in the peer storages have to be gathered to restore the owner's data. If there are many lost divided data in the distributed storage, the calculation time will be higher for regenerating the disconnected data by using the parity data to restore the owner's data. Furthermore, to get the owner's data, all separated data have to be gathered from the distributed storage.

2.3 The Advanced Encryption Standard

Advanced Encryption Standard(AES) [25], [26] is a symmetric key algorithm, which means the same key is used for both encrypting and decrypting the data. Several transformations are to be performed on data stored in an array. For this, the cipher is to put the data into an array after the cipher transformations are repeated over several encryption rounds. The number of rounds is determined by the key length, with 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. This encryption has been used in the storage scheme [27]. We use AES-256 for the proposed scheme to encrypt the divided data for the initiation algorithm to ensure security.

3. Proposed Scheme

In order to overcome the shortcomings mentioned in Sect. 2, we propose the server-based distributed storage using Secret Sharing with AES-256 for lightweight safety restoration. We reduce the burden of data capacity and the calculation time for the distributed storage by combining Secret Sharing with AES-256. AES-256 operates to ensure the security of the owner's divided data before converting the divided data by Secret Sharing. Secret Sharing and the server storage are utilized for the flexible restoring the owner's data. Secret Sharing operates that the owner can restore its data from the distributed storage with a few parity data. Furthermore, to safely keep the owner's data from the leaking of the pair of shares on Secret Sharing from the inner attack, the server storage, which is always ready to be connected, is used as the same with the distribute storage schemes using Reed Solomon [16]–[18]. The owner's data will be preserved in the distributed storage from the attacker in the proposed scheme. That is a new approach for the distributed storage scheme. This section is classified as six categories: 3.1. Encryption, Division, 3.2. Combination, Decryption, 3.3. Key Management, 3.4. Storing Process, 3.5. Restoring Process, and 3.6. Response Process.

3.1 Encryption, Division

In this part, to reduce the burden of data capacity and the calculation time for the distributed storage, we design the algorithms on AES-256 and Secret Sharing with the server storage. Before processing AES-256, we set that the owner's data are divided into two parts per rate to store each part into the peer storage and the server storage, respectively. According to [18], the security for the distributed storage is guaranteed regardless of the rates for dividing data. Authors denote that even if the server storage has 99% of the owner's original data, it is hard for the server-side to decrypt and restore the rest of the data. The comparisons of the performance per rates are described in Sect. 4. Therefore, we also denote that the owner's data can be restored from the process of Division, Encryption in AES-256 and Secret Sharing

Algorithm 1 Division in AES-256 and Secret Sharing

```
Encryption, Division;
Input: Original data.
Output: Encrypted pieced two data in AES-256.
$m$ = Original data;
$R_\alpha$ = Rate of $m$ for $a$, $R_\beta$ = Rate of $m$ for $b$;
$a$ = First part of $m*R_\alpha$, $b$ = Second part of $m*R_\beta$;
Divide $m$ into two pieces as $a$, $b$;
Encrypt $a$, $b$ in AES-256.
Secret Sharing;
Input: $b$ to secret share in $n$.
Output: Secret shares $s_1, s_2, s_3, \cdots, s_n$ of same size as $b$.
Set the number of share to restore in $i$
$s_1$ = Share 1 of $b$, $s_2$ = Share 2 of $b$, $s_3$ = Share 3 of $b$, $\cdots$, $s_n$ = Share $n$ of $b$;
Convert $b$ to $s_1, s_2, s_3, \cdots, s_n$ by Secret Sharing.
Division;
Input: $s_1, s_2, s_3, \cdots, s_n$ to divide.
Output: $s_{1p1}, s_{1p2}, s_{2p1}, s_{2p2}, s_{3p1}, s_{3p2}, \cdots, s_{np1}, s_{np2}$.
$s_1, s_2, s_3, \cdots, s_n$ in half into $s_{1p1}, s_{1p2}, s_{2p1}, s_{2p2}, s_{3p1}, s_{3p2}, \cdots, s_{np1}, s_{np2}$, respectively.
```

Algorithm 2 Restoration

```
Combination;
Input: $s_{2p1}, s_{2p2}, s_{3p1}, s_{3p2}$ from the peer storages.
Output: $s_2$, $s_3$.
Combine $s_{2p1}$ with $s_{2p2}$, $s_{3p1}$ with $s_{3p2}$ to be $s_1$, $s_2$, respectively
Decryption;
Input: $s_2$, $s_3$.
Output: $b$.
Convert $s_2$ with $s_3$ to be $b$ in $i$ on Secret Sharing
Input: $a$, $b$.
Output: $m$.
Decrypt $a$, $b$ in AES-256;
Restore $m$ by combining decrypted $a$ with decrypted $b$
```

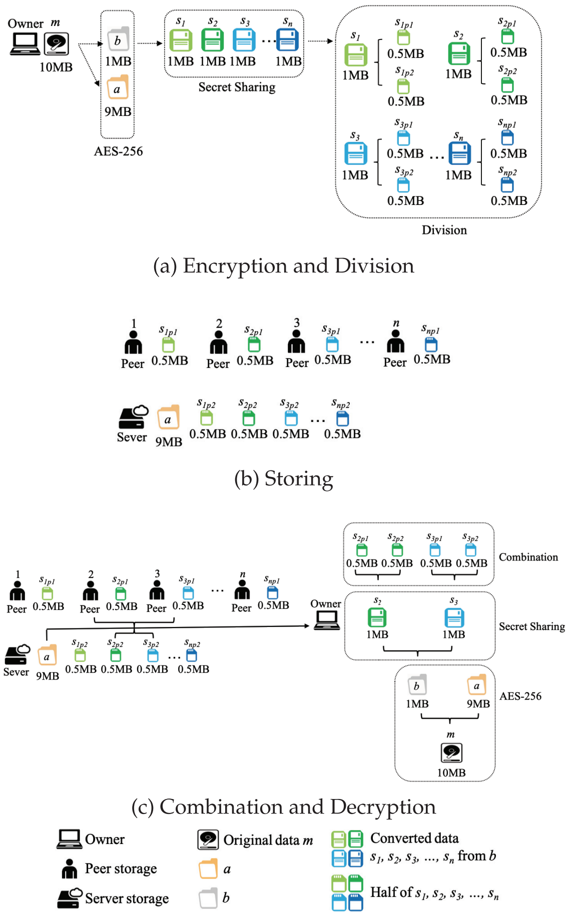


Fig. 3 Structure of Proposed Scheme

without any problem for regenerating the owner's data. Figure 3 shows the structure of the proposed scheme. There are the personal device, the peer storages, and the server storage for the proposed scheme. The size of the original data is set as 10MB to explain. Algorithm 1 shows the process of Encryption, Division. As shown in Algorithm 1, the original data of the owner m, which is the black hard disk drive icon in Fig. 3, is divided into the two parts as a, which is the orange file icon, and b, which is the gray file icon, per rates R_α, R_β in Fig. 3(a). The rates, R_α, R_β of the size of m are changeable by how much the owner wants to save its storage space. a, b are encrypted by AES-256 to keep secure the owner's data. b is converted by Secret Sharing within i, which is needed the parity number of share to restore b, when the distributed data and the peer storages are disconnected before b is stored into the peer storages. Here, if i set as two, b will be restored via Secret Sharing with two shares. To increase the value of i for restoring b, after b divided data are stored, divided b must be gathered and decrypted with AES-256. In addition, since the value of i is encrypted by AES-256, it cannot increase the value of i without AES-256. To guarantee the restoration, b is converted in Secret Sharing into parts as s_1 which is the yellow-green disk, s_2 which is the green disk, s_3 which is the sky-blue disk, $\cdots$, s_n which is the blue disk within n which is set the number of the shares by Secret Sharing. In that time, the data size of s_1, s_2, s_3, $\cdots$, s_n are set as the same as b by Secret Sharing. If the data size of b is 10MB and divided into four parts, each part has 10MB, respectively as the same with b. To preserve the original data from the leaking of pairs of shares on Secret Sharing, s_1, s_2, s_3, $\cdots$, s_n are divide into half into two parts, respectively, as s_{1p1}, s_{1p2}, s_{2p1}, s_{2p2}, s_{3p1}, s_{3p2}, $\cdots$, s_{np1}, s_{np2}. As shown in Fig. 3(b), the owner sends s_{1p1}, s_{2p1}, s_{3p1}, $\cdots$, s_{np1} into each peer storages. a, s_{1p2}, s_{2p2}, s_{3p2}, $\cdots$, s_{np2} are stored into the server storage by the owner. The separated data are safely kept in the distributed storage. Then, a is sent to the server storage to save the capacity of storage for peer storages. The attacker cannot seek where separated data are located since the server storage and the peer storages cannot know which storages have the pair of share to restore b. Moreover, even if the pair of share on Secre Sharing is leaked and the peer storage and the server storage have the part of the original data, the original data are securely kept since it is hard to restore and get the original data.

3.2 Combination, Decryption

This part shows how we reduce the calculation time when the owner restores its data from the distributed storage. The owner does not need to receive all of the divided stored data from the peer storages for the restoration. Moreover, the owner can restore its data with a few parity data from the leaking of peer storages. Here, to restore the original data m, the owner has to gather the divided stored data from the peer storages and the server storage. As shown in Algorithm 2 and Fig. 3(c), the owner just downloads s_{2p1} which is the half green disk of s_2, s_{3p1} which is the half sky-blue disk of s_3 from the peer storages and s_{2p2} which is pair data of s_{2p1}, s_{3p2} which is the pair data of s_{3p1}, a which is the orange file icon from the server storage to decrypt, respectively. Then, the owner combines s_{2p1}, s_{3p1} and s_{2p2}, s_{3p2} to be s_2 and s_3, respectively. If i is set as two, b will be restored by s_2 and s_3 in i via Secret Sharing. This means that only two parity data exist in the distributed storage, the owner's data can be restored regardless of when peer storages are disconnected. This ensures fast restoration with a few parity data. After restoring b, a and b are decrypted by AES-256. Then, the owner combines the decrypted a with the decrypted b to be the original data m. Therefore, the owner's data will be securely restored with a few parity data via the AES-256 with Secret Sharing.

3.3 Key Management

In the proposed scheme, the secret key of AES-256 is used to encrypt two separated data which are the a, b. Figure 4 shows the structure of key management. To safely restore and store the secret key regardless of leaking the peer storage, the secret key is converted by Secret Sharing into the pieces as k_1, k_2, k_3, $\cdots$, k_n as shown in Fig. 4. However, Secret Sharing method is weak from the inner attack by leaking the pairs of shares on Secret Sharing. To safely keep the secret key from the inner attack, k_1, k_2, k_3, $\cdots$, k_n are divided in half into two parts as k_{1p1}, k_{1p2}, and k_{2p1}, k_{2p2}, etc., as the process of division in Fig. 4. Then, those separated secret keys are stored into the peer storages and the server storage, respectively. The secret key is regenerated by gathering and combining the divided data of the secret key, such as k_{1p1}, k_{1p2}, and k_{2p1}, k_{2p2} only by the owner. In the proposed scheme, the attacker cannot get the secret key from the owner since the owner does not have the secret key. This is because the divided secret keys are stored in the peer storages and the server storage. Moreover, the attacker cannot restore the owner's secret key even if the pair of share on Secret Sharing is leaked since it is hard to know which store has the pair of share for restoring the secret key. Therefore, we can safely manage and keep the secret key.

3.4 Storing Process

As for the storing process for the owner's data, firstly the owner joins the distributed storage network as new peer storage. Then, the owner separates its data into the divided data. After the owner's divided data are encrypted by using AES-256, they are converted by Secret Sharing. To store them, the owner randomly selects the peer storages. The owner records which distributed storages store the pieced data to the hash table while simultaneously sending the pieced data. The hash table is located in the owner's device. Moreover, since the peer storages are randomly selected by the owner and just keep his/her data, they even do not know each connection of the peer storages. The peer storages and the server storage just hold the owner's divided data.

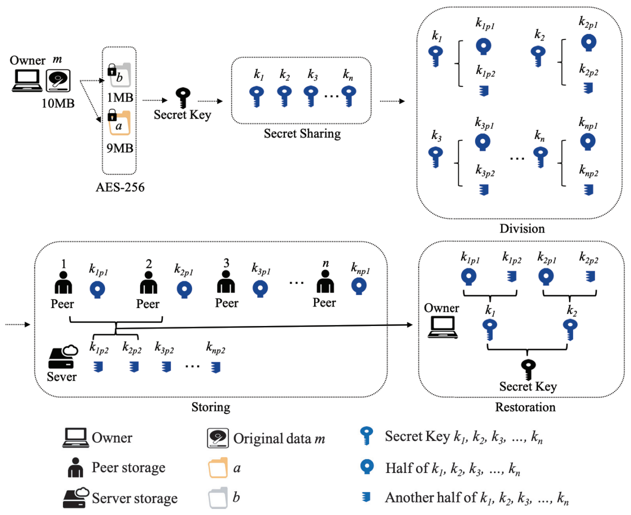

Owner m
10MB
b
1MB
a
9MB
AES-256
Secret Key
k_1 k_2 k_3 k_n
Secret Sharing
k_1 k_{1p1} k_{1p2} k_2 k_{2p1} k_{2p2}
k_3 k_{3p1} k_{3p2} k_n k_{np1} k_{np2}
Division
1 2 3 n
Peer
k_{1p1} k_{2p1} k_{3p1} k_{np1}
Sever
k_{1p2} k_{2p2} k_{3p2} k_{np2}
Storing
Owner
k_1 k_2
Secret Key
Restoration
Owner
Peer storage
Server storage
Original data m
a
b
Secret Key k_1, k_2, k_3, ..., k_n
Half of k_1, k_2, k_3, ..., k_n
Another half of k_1, k_2, k_3, ..., k_n

Fig. 4 Structure of Key Management

3.4.1 Owner

(1) Join to the distributed storage network as new peer storage.

(2) Divide the original data m into a, b.

(3) Encrypt a, b by using AES-256.

(4) Convert b into N shares s_1, s_2, s_3, $\cdots$, s_n by using Secret Sharing and Set the number i of shares to restore b.

(5) Divide s_1, s_2, s_3, $\cdots$, s_n in half into N pieces s_{1p1}, s_{1p2}, s_{2p1}, s_{2p2}, $\cdots$, s_{np1}, s_{np2}, respectively.

(6) Select the peer storages to store the divided data while recording the storage list to hash table

(7) Send s_{1p1}, s_{2p1}, $\cdots$, s_{np1} one by one to the peer storages.

(8) Send a, s_{1p2}, s_{2p2}, $\cdots$, s_{np2} to the server storage.

3.4.2 Peer storage

(1) Keep connecting the distributed storage network.

(2) Receive one by one in s_{1p1}, s_{2p1}, $\cdots$, s_{np1} from the owner.

3.4.3 Server Storage

(1) Keep connecting the distributed storage network.

(2) Receive a, s_{1p2}, s_{2p2}, $\cdots$, s_{np2} from the owner.

3.5 Restoring Process

As for when the owner wants to retrieve its data, firstly the owner checks the peer storages which are connected via the hash table. The owner only knows which pieces are needed to restore the original data. This is because the owner manages the pieced data and the divided secret keys, which are located in the distributed storage, to restore the original data by using the hash table. After the owner gathers its divided data from the peer storages, the owner restores one of the parts of its original data via Secret Sharing. The owner has to get its divided data as a priority from peer storages. This is because peer storages can be disconnected while the server storage is reliable. Then, the owner contacts the server storage to receives the rest of the parts of its original data. Finally, the owner's original data can be restored after the parts of its original data are decrypted via AES-256.

3.5.1 Owner

(1) Check the peer storages which are connected in the distributed storage network via the hash table.

(2) Receive s_{1p1}, s_{2p1} from the peer storages if the number i of shares is two to restore b.

(3) Receive a, s_{1p2}, s_{2p2} from the server storage.

(4) Combine s_{1p1}, s_{1p2}, s_{2p1}, s_{2p2} to s_1, s_2.

(5) Restore b by using s_1, s_2 via Secret Sharing.

(6) Decrypt a, b via AES-256.

(7) Restore the original data m by combining a, b.

3.5.2 Peer storage and Server storage

(1) Send the data which are requested to the owner.

3.6 Response Process

The peers just hold the owner's divided data until the owner requests its divided data to restore its original data from the peers. In addition, the owner does not keep tracking its data. The owner just tracks its data when the owner stores and restores its data in peer storages by using the hash table. This means that the owner does not keep checking how peer storages hold its data. However, the owner can check the status of stored data in three cases as below:

(1) When the owner stores its divided data to peers.

(2) When the owner requests for its data to peers.

(3) When the owner checks how the peers hold its data.

As for the case (1), the peers respond to the owner after they hold the owner's divided data. As for the case (2), the peers respond to how they hold the owner's data to the owner when the owner wants to gather its divided data to restore the original data from peers. As for the case (3), the peers respond like the case (2) when the owner checks its data which is located in the peers by using its hash table.

Table 1 Experiments Environment

Items	Parameter value
Operating System	Ubuntu Linux 16.04.3 LTS
Programming language	Python
CPU	Intel i9-7900X 3.38Ghz
Memory	64GB
Data Size	1GB
Number of the peer storages	100
Number of the parity share	2

Table 2 Calculation time in AES-256 per rates on 1GB

Case	Rate	Encryption [SEC]	Decryption [SEC]	Total [SEC]
1	a:80% / b:20%	11	11	22
2	a:82% / b:18%	11	11	22
3	a:84% / b:16%	11	11	22
4	a:86% / b:14%	11	11	22
5	a:88% / b:12%	11	10	21
6	a:90% / b:10%	10	10	20
7	a:92% / b:8%	10	10	20
8	a:94% / b:6%	10	10	20
9	a:96% / b:4%	9	10	19
10	a:98% / b:2%	9	10	19

Table 3 Calculation time in Secret Sharing on b

Case	Rate	Encryption [SEC]	Decryption [SEC]	Total [SEC]
1	b:20%	0.0033	0.0011	0.0044
2	b:18%	0.0030	0.0009	0.0039
3	b:16%	0.0026	0.0009	0.0036
4	b:14%	0.0024	0.0008	0.0032
5	b:12%	0.0019	0.0006	0.0025
6	b:10%	0.0017	0.0005	0.0022
7	b:8%	0.0013	0.0004	0.0017
8	b:6%	0.0010	0.0003	0.0013
9	b:4%	0.0007	0.0003	0.0010
10	b:2%	0.0004	0.0002	0.0006

4. Results and Evaluations

We compare the proposed scheme with the conventional distributed storage scheme using Reed Solomon and Secret Sharing in terms of lightweight, stability, and safety. The lightweight part describes the comparison data size and the calculation time. The stability part shows the efficiency for restoring the owner's data from the distributed storage. The safety part explains how our proposed scheme guarantees the security for the owner's data.

4.1 Lightweight

Here, the evaluations of the lightweight are classified into

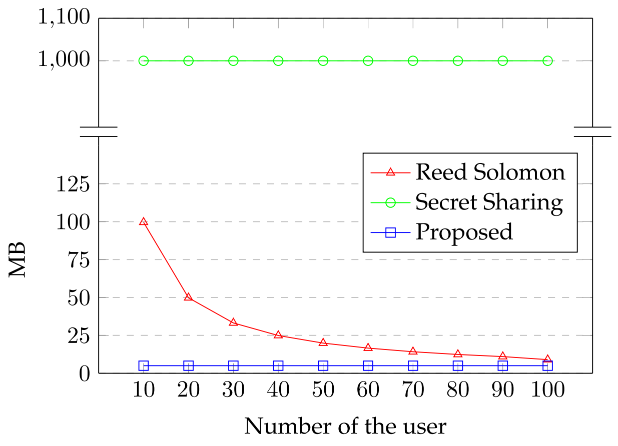


Fig. 5 Comparison of the data size on a single peer

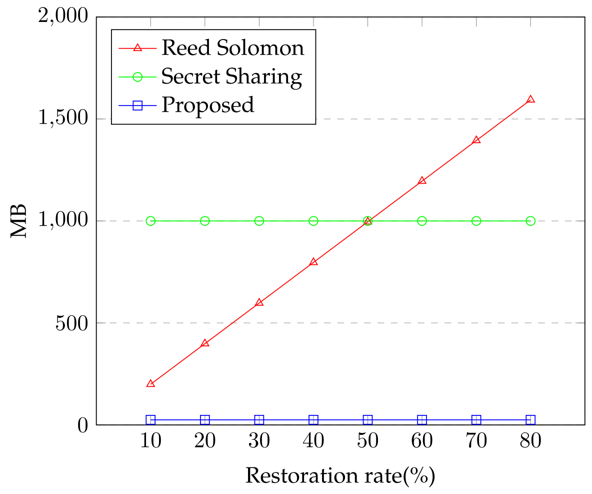


Fig. 6 Comparison of the data size of the parity data

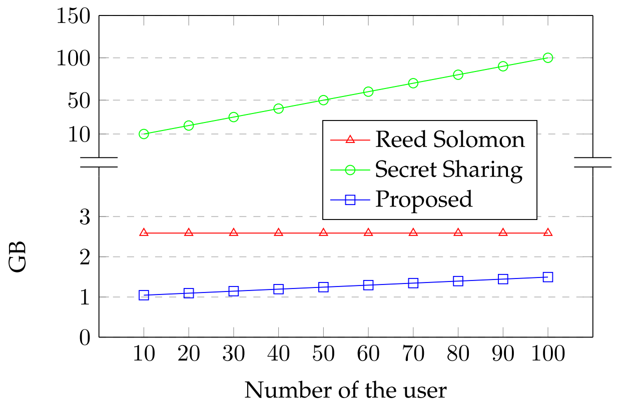


Fig. 7 Comparison of the total data size

two points: Reduction of the burden of data capacity, Reduction of the calculation time in the environment of the experiments are shown in Table 1. To evaluate, we need to clearly set the rates for dividing the original data since there is no suitable rate of data for the peer storage and the server storage in the conventional distributed storage schemes. To clearly set the rate of data, we compare the performance in the situation where the rates for dividing the original data changes from 80%, 20% to 98%, 2%. Table 2 shows the calculation time in AES-256 per rate. As we can see from Table 2, as a is larger, the calculation time becomes slightly less since the process for dividing the original data into a, b are light. However, there is no big difference in the calculation time in 10 cases since the rate of data has been less impact on the process of AES-256. The original data are restored from a, b without any problem for regenerating the original data. Table 3 shows the comparison of the calculation time in Secret Sharing for the owner. As we can see from Table 3, the calculation time of Secret Sharing for b is lesser as the size of b is smaller. This is because the burden of the operation on Secret Sharing for the owner is low. As for the following simulation, we need to set a suitable rate for dividing original data. Each rate for a, b can be changeable by the owner since the impact by changing the rate on the security is less. Therefore, we use the middle rates for a, b as 90%, 10% respectively to evaluate the performances between the proposed scheme and the conventional schemes.

4.1.1 Reduction of the Burden of Data Capacity

$$Single_{Reed} = f_{Reed}(m) = \frac{m}{n} \tag{1}$$

$$Single_{Secret} = f_{Secret}(m) = m' \tag{2}$$

$$a = \alpha * m,\ b = (1-\alpha) * m,\ f_{Secret}(b) = c \tag{3}$$

$$Parity = (2 * n) * \beta = \rho \tag{4}$$

$$Single_{Proposed} = \frac{c}{2} = d \tag{5}$$

$$Total_{Reed} = (\frac{m}{n} * n) + (\frac{m}{n} * \rho) \tag{6}$$

$$Total_{Secret} = m' * n \tag{7}$$

$$Total_{Proposed} = (a + c * n) + (c * n) \tag{8}$$

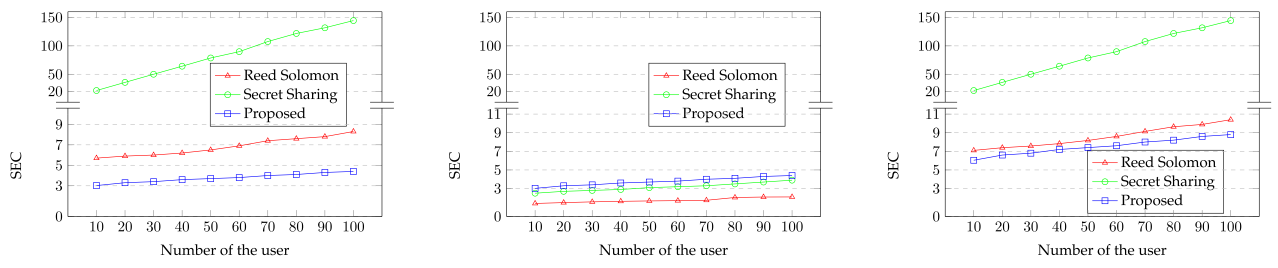
This section explains how we reduce the burden of data capacity. Figure 5 shows the comparison of data size on the single peer for the distributed storages from 10 peers to 100 peers. m, n, denote the original data 1GB and the number of peer storages, respectively. In the conventional distributed storage using Reed Solomon, m is divided as n as shown in (1) through Reed Solomon operation. The peer storages in the distributed storages have the same size of divided data via Reed Solomon. The data size on the single peer in the distributed storage using Reed Solomon is decreasing with the increasing the number of the peer storages in (1). In the conventional distributed storage using the Secreting Sharing, the owner's original data are converted as the number of peer storage by Secret Sharing. m' denotes the converted data size by Secret Sharing in (2). Since the divided data size as the same with the owner's original 1GB data, the data size is much larger. Every peer storage in the distributed storage stores 1GB data regardless of the increasing the peer storages via Secret Sharing. So, the size of data which the peer stores is much larger as shown in Fig. 5. However, in the proposed scheme, the data size which the peer will store is much smaller even if we adapt Secret Sharing as shown in Fig. 5. Where β in (3), ρ in (4) denote the size of parity data, the percentage for restoration, respectively. m is separated into a, b by using AES-256. α denotes the percentage of data size for a 90% of m, b 10% of m. Through Secret Sharing in (3), the shared data c from b have the same size as b. After the owner divides c into two parts as d in (5), d will be stored in the single peer storage in the proposed distributed storage. The peer only stores d 5% data from the original data in the proposed scheme. So, the owner's burden of the capacity can be reduced in the proposed scheme. Figure 6 shows the comparison of the parity data size while reflecting the restoration percentage from 10% to 80% in 100 peer storages. Here, we assume that the owner's 1GB data are divided into 100 pieces and stored in 100 peer storages, respectively. In the distributed storage using Reed Solomon, the parity data size is increasing as the restoration rate is growing. The parity data size in the distributed storage using Reed Solomon for 80% restoration is 1.593MB. In the conventional Secret Sharing storage and the proposed scheme, the parity data size is the same and fixed via Secret Sharing regardless of the restoration rate. Moreover, since two parity data are set to restore the owner's original data, the restoration rate is guaranteed 80% more even though the owner's data 8 divided data are disconnected from 10 peer storages. This means that the proposed scheme ensures the higher restoration than the distributed storage using Reed Solomon. In the conventional Secret Sharing storage, the size of parity data is much larger since 1GB data is converted via Secret Sharing method. The parity data size in the conventional Secret Sharing storage for 80% restoration is 1GB. However, in the proposed scheme, the size of parity data is smaller since only 10% of 1GB data is converted on Secret Sharing method and divided into two pieces which the half of 10%. The parity data size in the proposed scheme for 80% restoration is 25MB. Figure 7 shows the comparison of the total data size on the distributed storage for the restoration rate of 80% from 10 peers to 100 peers. The total data size of the conventional distributed storage using Reed Solomon is the same regardless of increasing the number of the peers since Reed Solomon code makes the size of data to be equal in (6). The total data size of the conventional Secret Sharing storage is growing in (7) by raising the restoration rate and the number of the peers in the distributed storage. In (8), the total data size in the proposed scheme is much smaller than the convention schemes since a few pari ty data are needed for the restoration. This shows that our proposed scheme can reduce the burden of data capacity for the distributed storage.

4.1.2 Reduction of Calculation Time

Here, to compare the calculation time, we evaluate the calculation time when the owner divides, encrypts and sends the data to peers storages from 10 to 100. Figure 8 shows the comparison of the calculation time without the restoration. As for the sending and encryption time in Fig. 8(a), the calculation time of the conventional Secret Sharing storage is much higher than the distributed storage using Reed Solomon and the proposed scheme since the process for converting 1GB via Secret Sharing takes more time. However, the calculation time of our proposed scheme is low since only 10% of 1GB data is converted via Secret Sharing. As shown in Fig. 8(b), the calculation time of the proposed scheme is a bit higher since the owner has to receive the data from server storage and peer storages regardless of the restoration. The distributed storage using Reed Solomon and the conventional Secret Sharing download the data only from peer storages if there is no restoration about the owner's data. Additionally, the decryption with AES-256 in the proposed scheme takes more time. However, the total calculation time of our proposed scheme is low as shown in Fig. 8(c). Figure 9 shows the comparison of the calculation time of receiving and decryption with the restoration. We evaluate the receiving and decryption time in cases of restoration 10%, 50%, 100% from 10 peer storages to 100 peer storages. The calculation time of the distributed storage using Reed Solomon is higher in Fig. 9(a) since the operation of Reed Solomon takes more time to restore the disconnected data. From the restoration of 50% disconnected data in Fig. 9(b), the calculation time of the distributed storage using Reed Solomon is increasing rapidly since the process of regenerating the disconnected data takes more time. Finally, from restoration 80% disconnected data in Fig. 9(c), the calculation time of the distributed storage using Reed Solomon is highest. The calculation time of the distributed storage using Reed Solomon is increasing and much higher with the restoration rate since the operation of Reed Solomon needs many parity data to restore disconnected data. On the other hand, the calculation time of the proposed scheme and the conventional Secret Sharing storage is lower regardless of the restoration rate. This is because, in both schemes, the disconnected data can be restored by Secret Sharing with a few parity data. As shown in Fig. 9, the calculation time of the proposed scheme is a bit higher than the conventional Secret Sharing storage since the decryption with AES-256 takes additional time to restore data in the proposed scheme. However, as shown in Fig. 10, the total calculation time of our proposed scheme is much lesser than the other schemes. This shows that our proposed scheme can reduce the calculation time for the distributed storage.

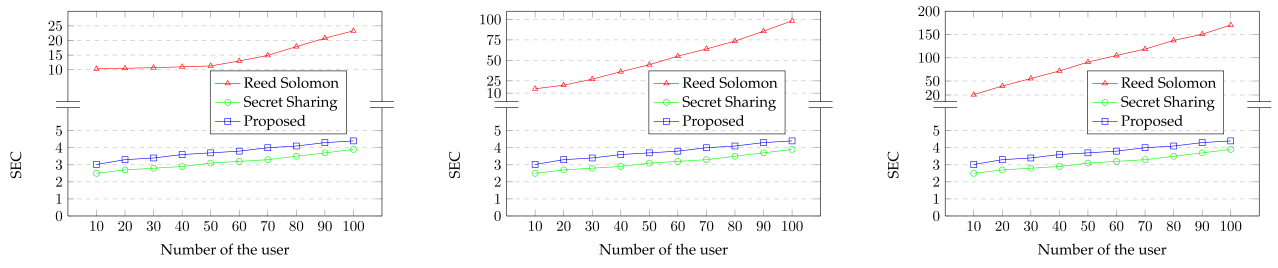
4.2 Stability

The proposed scheme shows the stability in two categories



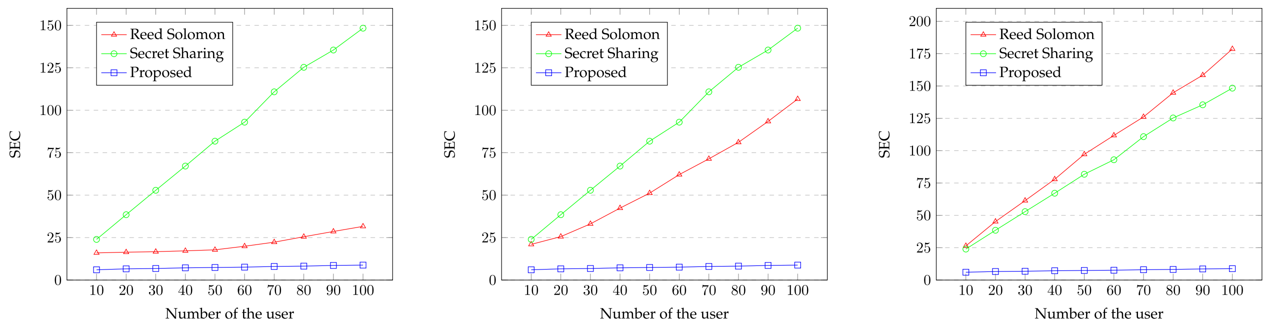
(a) Sending - Encryption (b) Receiving - Decryption (c) Total

Fig. 8 Comparison of the calculation time without the restoration



(a) Restoration 10% disconnected data (b) Restoration 50% disconnected data (c) Restoration 80% disconnected data

Fig. 9 Comparison of the calculation time of Receiving - Decryption with the restoration



(a) Restoration 10% disconnected data (b) Restoration 50% disconnected data (c) Restoration 80% disconnected data

Fig. 10 Comparison of total calculation time with the restoration

as Restoration with a few data and Restoration from disconnected peer storage.

4.2.1 Restoration with a Few Data

In the distributed storage using Reed Solomon, in order to restore the stored data in the peer storage, the parity data are mandatory. This means that the parity data will be increased by how much the parity data can recover the disconnected data. For example, when the count of the pieced stored data are 10, if one of 10 is lost, two parity data are needed to restore the original data. Moreover, to regenerate the original data, the owner has to gather all pieced data from the distributed storage. That is, if the owner finds that some pieced data are disconnected, the original data will be restored after the disconnected data are repaired by the parity data. However, our proposed scheme is more flexible than the distributed storage using Reed Solomon. The owner does not need to gather all pieced data and regenerate the lost divided data from the distributed storage to get the original data in the proposed scheme. For example, if the count of the pieced stored data are 10 and the parity data are set as 2 to restore the original data, even if 8 pieced data are disconnected, the original data will be restored by using only 2 parity data through Secret Sharing. This means that the owner's data will be restored with a few parity data from the distributed storage.

4.2.2 Restoration from Disconnected Peer Storage

The owner's data can be restored although there are the disconnections from the peer storages. This is because Secret Sharing in the proposed scheme ensures the restoration of the owner's data from the disconnections of peer storages as we mentioned in 4.2.1. If all peers do not respond when the owner requests for gathering its data to restore original data,

Table 4 Synthesis results

Scheme	Dependency on the server storage	Burden of the data capacity	Volume of the parity data	Calculation time without restoration	Calculation time with restoration
Reed Solomon	High	Mid	High	Low	High
Secret Sharing	Low	High	Mid	High	Low
Proposed	High	Low	Low	Low	Low

the owner's data will be lost. There is no way to restore the owner's data from lost all peers. However, according to [31], all peers are offline at the same time is less. The owner's data can be restored when the least peers are back.

4.3 Safety

Here, we explain how we guarantee the safety of the owner's data in the distributed storage scheme. In the conventional Secret Sharing storage, if the pair of data are leaked, the divided data might be restored. This is a serious problem for safety. When 10 separated converted data are stored in the peer storages, if two or more peers know the pair of stored data, those peers can get the owner's original one via the Secret Sharing. The Secret Sharing method is weak from the inner attack. In the distributed storage using Reed Solomon, the server can get the original data by using the parity data since the parity data are located in the server storage. To overcome those problems and ensure the safety of the owner's data in the proposed scheme, the pair of shares is divided into half to protect the owner's data from leaking of the pair of shares on Secret Sharing. Half of the pair of shares is encrypted by AES-256. Those shares are stored in the peer storages and server storage. Moreover, a secret key for AES-256 is also divided and stored only by the owner respectively. The owner only knows the connections of shares in the peer storages and server storage via its hash table. Therefore, the attacker will experience difficulty in getting the owner's data even if it knows the pair of shares on Secret Sharing. Moreover, although the server acts as the storage like the conventional Reed Solomon storage in the proposed scheme, the server cannot know and find where, which peer storage has the divided data as same as the attacker. Furthermore, in the proposed scheme, even if the server storage has the 90% of the owner's encrypted data on AES-256 and the half of divided data on the Secret Sharing, the server cannot restore the owner's original data. Similarly, the peer storages cannot regenerate the data since they only have a half like the server storage. Although the server storage and the Secret Sharing has the problem from the leaking of data pairs are adapted to the proposed scheme, the owner's data can be safely stored.

5. Discussion and Limitation

We research how to improve the performance of the owner in the distributed storage scheme. Table 4 shows the synthesis results. Even if the proposed scheme shows better performance than the conventional storage schemes, there are limitations in the proposed scheme.

5.1 Threat of the Attack to the Server Storage

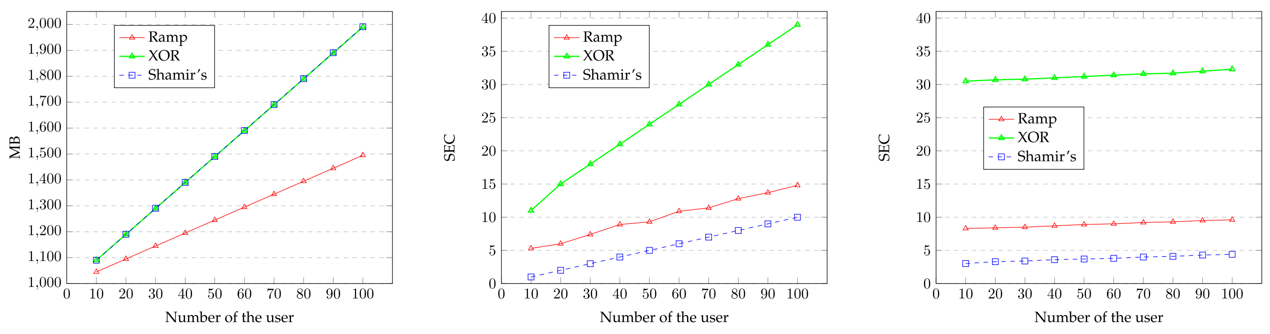
We assume that there is no absolutely secure server storage. Therefore, to protect the owner's data is important in the distributed storage with server storage scheme. For this, the owner's data are divided and encrypted by Secret Sharing and AES-256 in the proposed scheme. However, there might be a problem in the proposed scheme since the server storage is utilized. As shown in Table 4, the proposed scheme and the distributed storage using Reed Solomon are dependent on the server storage since the parity data for restoring the owner's data are stored in the server storage in both schemes. The server storage is reliable to reduce the burden of data capacity on the peer storage. If the server storage is attacked, the parity data for the peer might be lost. So, we should discuss the threat of the attack on the server storage. we can consider three cases of the attack scenarios for the server storage as below:

(1) The stored data in the server storage are leaked by the manager of the server.

(2) The stored data in the server storage are threatened by the attacker.

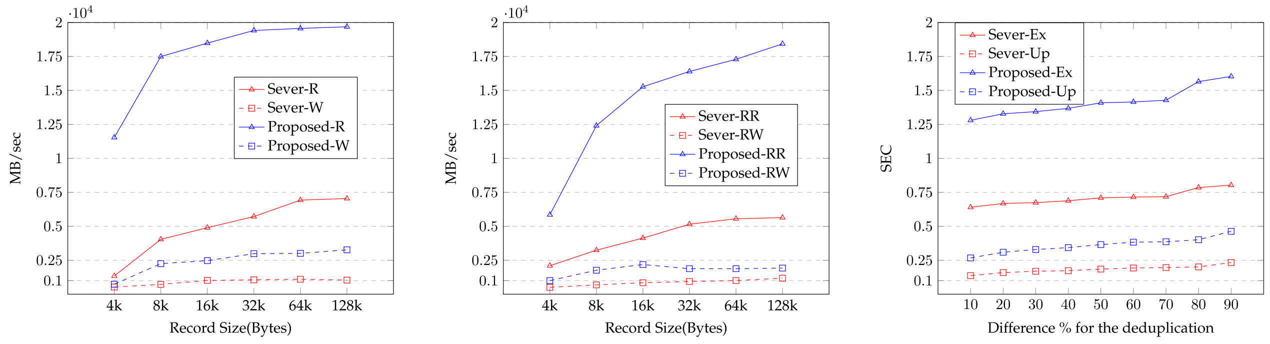
(3) The server storage is downed.

As for the case (1), the manager of the server might get the original data although the original data are encrypted. Since the manager has the whole data, the manager can threaten the original data in various ways such as the brute force. However, in the proposed scheme, the manager cannot deal with the original data since the original data are divided and stored in the peer storages. As for the case (2), the attacker might get the original data like the manager of the server in case (1). However, the attacker cannot get the original data for the same reason with the case (1). Moreover, in the proposed scheme, the attacker is hard to seek where the divided data are stored into peer storages. The secret key for the original data are also separated into peer storages. Therefore, the attacker cannot get and gather the original data. As for the case (3), if the server storage is downed, the owner cannot get original data in the centralized storage and the distributed storage with the server storage including the proposed scheme and the conventional scheme. This is because the server storage has the parts of the original data and the parity data for restoring data in the peer storages are disconnected. This is the same problem for both schemes. The two attack scenarios for the server storage can be solved in the proposed scheme. Therefore, the owner's data will be kept safe in the proposed scheme although the server storage



(a) Total data size (b) Sending - Encryption time (c) Restoration - Decryption time

Fig. 11 Comparison of the performances for Secret Sharing



(a) Read - Write (b) Random Read - Random Write (c) Deduplication

Fig. 12 Comparison of the performances for the storage service

might be threatened.

5.2 Comparison with Other Secret Sharing

To evaluate Shamir Secret Sharing in the proposed scheme in more various aspects, we compared the Shamir's Secret Sharing with Ramp Secret Sharing [28] and XOR Secret Sharing [29] from 10 peers to 100 peers in terms of the total data size, the calculation time of sending-encryption, the calculation time of restoration-decryption. Figure 11 shows the comparisons of the performances for Secret Sharing. Figure 11(a) shows the comparison of total data size. The total data size of Ramp Secret Sharing is less than half XOR Secreting Sharing and Shamir's Secret Sharing since Ramp Secret Sharing sets the size of share on Secret Sharing to be less by grouping each share as the participants. However, the size of share is big in XOR Secret Sharing and Shamir's Secret Sharing since each share has the same size as the original data. Therefore, Ramp Secret Sharing might be suitable for lightweight distributed storage. Figure 11(b) and Fig. 11(c) show that the comparisons of the calculation time for sending and restoring the data from peer storages in Shamir's Secret Sharing in the proposed scheme is much lesser than Ramp Secret Sharing and XOR Secret Sharing since it takes more calculation time to encrypt and decrypt the share from the groups of share and peer storages in Ramp Secret Sharing. This is because the process of XOR Secret Sharing is heavy to compare the binary for XOR operation, the calculation time for encrypting and decrypting much higher. By contrast, Shamir's Secret Sharing in the proposed scheme shows better performance. For the future works, we will deal with Ramp Secret Sharing to reduce the calculation time for distributed storage.

5.3 Comparison of the Performances with the Server Storage

To evaluate the performances of the proposed scheme with the server storage, we evaluated the process by using IOzone [30] from 4k record size to 128k record size for 1GB. Figure 12 shows the comparison of performances for the storage service. In Fig. 12(a), R and W denote the sequential read and sequential write, respectively. RR and RW denote the random read and the random write, respectively in Fig. 12(b). As shown in Fig. 12(a) and Fig. 12(b), the proposed scheme costs are higher than the server storage. The distributed storages occur higher overhead to read and write the data than the server storage. Because unlike the single server storage, the owner must access many peer storages. This means that distributed storage might be heavy for scalability. To evaluate the deduplication of the proposed scheme, we used a source-based deduplication method [32]. In the source-based deduplication method, the client has to make the difference between the original data and the changed data with the deduplication. To deduplicate the data in the storage, the client sends the patch file to the storage. Then, the storage updates the data to be changed by using the patch file. We compared the calculation time for the deduplication of the proposed scheme with the single sever storage method. We assume that there is a different rate from 10% to 90% for the deduplication. There are 100 peer storages in the proposed scheme. Figure 12(c) shows the comparison of the calculation time for the deduplication. In Fig. 12(c), Ex and Up denote the extraction of the difference from data and updating the data, respectively. As shown in Fig. 12, the calculation time of the proposed scheme are higher than the single server storage. Because, unlike the single server storage, the operation of the deduplication for 100 peer storage by the owner takes more time. Therefore, for these reasons, although the distributed storage can solve the problems of the server storage, the performances might be low for the storage service.

5.4 Incentive and Payment

As for the incentive, the proposed scheme can be applied to the commercial distributed storage services such as Maidsafe [13] and Storj [17]. The reason is that the peer, which just provides its storage without saving its data to other peers, exists in the proposed scheme. The peer can expect the incentive. According to [13], [17], the peer can get money from the operator while providing its storage for the commercial distributed storage service. As for the payment, the owner does not pay anything to use in the proposed scheme. However, in the commercial distributed storage service [13], [17], the owner pays to the operator for using storage service. Therefore, if the proposed scheme is used for commercial service, the owner needs to pay to use the storage service.

6. Conclusion

We have proposed the server-based distributed storage using Secret Sharing with AES-256 for lightweight safety restoration. The evaluations show that our proposed scheme has three advantages for the lightweight, the stability and the safety. The burden of data capacity and the calculation time can be reduced for the lightweight. As for the stability, the owner's data will be flexibly restored with a few parity data when the peer storages are disconnected. In the safety part, the owner's data will be securely kept although Secret Sharing method has the weakness of leaking the pair of data. Therefore, the owner can securely store its data in our proposed scheme.

Acknowledgments

This work is partly supported by the Grant in Aid for Scientific Research (No.17K06440) from Japan Society for Promotion of Science (JSPS).

References

[1] M. Chiang and T. Zhang, "Fog and IoT: An Overview of Research Opportunities," IEEE Internet Things J., vol.3, no.6, pp.854, Dec. 2016.

[2] D. Quick and K.K.R. Choo, "Google drive: Forensic analysis of data remnants," J. Network and Computer Applications, vol.40, pp.179, ELSEVIER, April 2014.

[3] I. Drago, M. Mellia, M.M. Munafo, A. Sperotto, R. Sadre, and A. Pras, "Inside Dropbox: Understanding personal cloud storage services," Proc. 2012 ACM Conference on Internet Measurement Conference - IMC '12, pp.481, Nov. 2012.

[4] D. Quick and K.K.R. Choo, "Digital Droplets: Microsoft SkyDrive Forensic Data Remnants," Future Generation Computer Systems, vol.29, no.6, pp.1378, Aug. 2013.

[5] C. Wang,, S.S.M. Chow, Q. Wang,, K. Ren, and W. Lou, "Privacy Preserving Public Auditing for Secure Cloud Storage," IEEE Trans. Comput., pp.62, no.2, pp.362–375, Feb. 2011.

[6] L. Xiao,, D. Xu,, C. Xie, N.B. Mandayam, and H.V. Poor, "Cloud Storage Defense Against Advanced Persistent Threats: A Prospect Theoretic Study," IEEE J. Selected Areas in Communications, vol.35, no.3, pp.534–544, March 2017.

[7] M. Du,, Q. Wang,, M. He,, and J. Weng, "Privacy-Preserving Indexing and Query Processing for Secure Dynamic Cloud Storage," IEEE Trans. Inf. Forensics Security, vol.13, no.9, pp.2320–2332, Sept. 2018.

[8] J. Li, J. Wu, and L. Chen, "Block-Secure: Blockchain Based Scheme for Secure P2P Cloud Storage," Information Sciences, vol.465, pp.219–231, Oct. 2018.

[9] B. Cui, Z. Liu, and L. Wang, "Key-Aggregate Searchable Encryption (KASE) for Group Data Sharing via Cloud Storage," IEEE Trans. Comput., vol.65, no.8, pp.2374–2385, Aug. 2016.

[10] "FBI asks Apple for help cracking Pensacola gunman's iPhones," Available: https://www.washingtonpost.com/national-security/fbi-asks-apple-for-help-cracking-pensacola-gunmans-iphones/2020/01/07/b829ac72-3178-11ea-91fd-82d4e04a3fac_story.html [Internet]

[11] "Legal process for user data requests FAQs," Available: https://support.google.com/transparencyreport/answer/7381738?hl=en [Internet]

[12] B. Tomas and B. Vuksic, "Peer to peer distributed storage and computing cloud system," 34th Int. Conf. Information Technology Interfaces, pp.79, IEEE, 2012.

[13] F. Jacob, J. Mittag, and H. Hartenstein, "A security analysis of the emerging P2P-based personal cloud platform maidsafe," 14th Int. Conf. Trust, Security and Privacy in Computing and Communications, pp.1403, IEEE, 2015.

[14] F. Rizzo, G.L. Spoto, P. Brizzi, D. Bonino, G.D. Bella, P. Castrogiovanni, Istituto Superiore, and Mario Boella, "Beekup: A distributed and safe P2P storage framework for ioe applications," 20th Conference on Innovations in Clouds, Internet and Networks, pp.44, IEEE, 2017.

[15] E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," IEEE Commun. Surveys Tuts., vol.7, no.2, pp.72–93, 2005.

[16] J.S. Plank and L. Xu, "Optimizing cauchy reed-solomon codes for fault-tolerant network storage applications," IEEE Interantional Symposium on Network Computing and Apllications, pp.173, 2006.

[17] S. Wilkinson, "Storj a peer-to-peer cloud storage network files as encrypted shards," StorJ, 2014, pp.1. Available: https://storj.io/storj.pdf [Internet]

[18] T. Wangian, J. Zhou, X. Chen, G. Wang, A. Liu, and Y. Liu, "A three-layer privacy preserving cloud storage scheme based on computational intelligence in fog computing," IEEE Trans. Emerging Topics in Computational Intelligence, vol.2, no.1, pp.3, Feb. 2018.

[19] I.S. Reed and G. Solomon, "Polynomial codes over certain finite

fields," Soc. Indust, vol.8, no.2, pp.300, June 1960.
[20] Q. Wang, J. Jing, and J. Lin, "A secure storage system combining secret sharing schemes and Byzantine quorum mechanisms," 10th IEEE Int. Conf. Computer and Information Technology, pp.596, 2010.
[21] M. Fukumitsu, S. Hasegawa, J. Iwazaki, M. Sakai, and D. Takahashi, "A proposal of a secure P2P-type storage scheme by using the Secret Sharing and the blockchain," IEEE 31st Int. Conf. Advanced Information Networking and Applications, pp.803, 2017.
[22] R.K. Raman and L.R. Varshney, "Distributed storage meets secret sharing on the blockchain," Information Theory and Applications Workshop, pp.1, IEEE, 2018.
[23] A. Shamir, "How to share a secret," Algorithms Unplugged, pp.159, 2011.
[24] R.L. Rivest, "All-or-Nothing encryption and the package transform," Springer Fast Software Encryption, vol.1267, pp.210–218 1997.
[25] J Daemen and V Rijmen, "The design of Rijndael: AES-the advancedencryption standard," Springer, 2013.
[26] A. Biryukov and D. Khovratovich, "Related-key cryptanalysis of the full AES-192 and AES-256," Lect. Notes Comput. Sci. (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol.5912 LNCS, pp.1, 2009.
[27] V.R. Pancholi and B. Patel, "Enhancement of Cloud Computing Security with Secure Data Storage using AES," International Journal for Innovative Research in Science & Technology (IJIRST), 2(09), pp.18, 2016.
[28] G.R. Blakley and C. Meadows, "Security of Ramp Schemes," CRYPT 1984, pp.242–268, 1984.
[29] A.K. Chattopadhyay, D. Ghosh, P. Maitra, A. Nag, and H.N. Saha, "A Verifiable (n,n) Secret Image Sharing Scheme Using XOR Operations," 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), pp.1025–1031, 2018.
[30] IOzone "Filesystem Benchmark," Available: http://www.iozone.org [Internet]
[31] K. Nils, "Secure Volunteer Computing for Distributed Cryptanalysis," Kassel university press, 2018, 260 Pages.
[32] S. Lee and D. Choi, "Privacy-preserving cross-user source-based data deduplication in cloud storage," Int. Conf. ICT Convergence, pp.329–330, 2012.

Sanghun Choi received his M.Sc. degrees from Keio University in 2018. He is a Ph.D. student at Keio University. His research interest is a security and privacy for the location system, the cloud system and IoT. He is a member of IEICE and IEEE.

Shuichiro Haruta was born in Saitama, Japan in 1992. He received his B.E., M.E., and Ph.D. (Engineering) degrees in Department of Information and Computer Science, Keio University, Yokohama, Japan in 2015, 2017 and 2020, respectively. His research interest is security privacy for Internet of Things. He was a research associate at Keio University (2017–2018).

Yichen An received his M.Sc. degrees from Keio University in 2020. His research interest is an information security. He is a member of IEICE.

Iwao Sasase was born in Osaka, Japan in 1956. He received the B.E., M.E., and D.Eng. degrees in Electrical Engineering from Keio University, Yokohama, Japan, in 1979, 1981 and 1984, respectively. From 1984 to 1986, he was a Post Doctoral Fellow and Lecturer of Electri- cal Engineering at the University of Ottawa, ON, Canada. He is currently a Professor of Information and Computer Science at Keio University, Yokohama, Japan. His research interests include modulation and coding, broadband mobile and wireless communications, optical communications, communication networks and information theory. He has authored more than 295 journal papers and 440 international conference papers. He granted 45 Ph.D. degrees to his students in the above field. Dr. Sasase received the 1984 IEEE Communications Society (ComSoc) Student Paper Award (Region 10), 1986 Inoue Memorial Young Engineer Award, 1988 Hiroshi Ando Memorial Young Engineer Award, 1988 Shinohara Memorial Young Engineer Award, 1996 Institute of Electronics, Information, and Communication Engineers (IEICE) of Japan Switching System Technical Group Best Paper Award, and WPMC 2008 Best Paper Award. He is now serving as a Vice-President of IEICE. He served as President of the IEICE Communications Society (2012–2014). He was Board of Governors Member-at-Large (2010–2012), Japan Chapter Chair (2011–2012), Director of the Asia Pacific Region (2004–2005), Chair of the Satellite and Space Communications Technical Committee (2000–2002) of IEEE ComSoc., Vice President of the Communications Society (2004–2006), Chair of the Network System Technical Committee (2004–2006), Chair of the Communication System Technical Committee (2002–2004) of the IEICE Communications Society, Director of the Society of Information Theory and Its Applications in Japan (2001–2002). He is Fellow of IEICE, and Senior Member of IEEE, Member of the Information Processing Society of Japan.