International Journal of Grid and Distributed Computing
Vol. 10, No. 11 (2017), pp.57-70
http://dx.doi.org/10.14257/ijgdc.2017.10.11.06

Efficiency Enhanced Combinatorial-based File Sharing Scheme for Distributed Storage with Heterogeneous Personal Devices

Jung-Eun Park and Young-Hoon Park*

Division of Computer Science, Sookmyung Women's University
je.park@sookmyung.ac.kr and yh.park@sookmyung.ac.kr

Abstract

Distributed storage plays an important role in the network of personal devices, owing to its fault tolerance and quick retrieval of stored files. To enhance the security and privacy of the stored data, secret sharing schemes have been employed for distributed storage. Among the existing secret sharing schemes, a combinatorial-based file sharing scheme is the most suitable one because of its lightweight structure and low cost. However, there remains the problem that the sizes of the distributed stored file shares in the devices do not support the heterogeneous personal device environment, which may cause an additional efficiency problem.

In this paper, we provide an efficiency-enhanced combinatorial-based file sharing scheme for distributed storage with personal devices, which considers the heterogeneous characteristics thereof. In addition to existing combinatorial-based file sharing, for the file storing process, we propose a method of determining the sizes of file shares according to the remaining storage capacities and average communication speeds of the participant personal devices. In the discussion, we demonstrate that our scheme manages the distributed storage system with personal devices efficiently compared to alternative existing sharing schemes.

Keywords*: distributed cloud storage; secret sharing; security; file sharing; cloud security*

1. Introduction

Distributed storage systems are receiving considerable attention owing to their fault tolerance and quick retrieval of stored files [2]. Initially, distributed storage systems consisted only of servers and PCs; however, as IT advanced, mobile devices such as smart phones and tablet PCs were developed. Moreover, various wearable devices such as smart watches, smart shoes, and smart glasses are expected to emerge to realize a "life revolution" by attaching a computer to a body in the near future. All these mobile devices will be connected to networks and will form part of a distributed storage system [3].

In these systems, when a file is saved, it is copied and stored to each storage. Because the original file is copied and stored to each storage, several security problems can occur. For example, if a distributed storage device is lost or stolen, the user's confidentiality may be breached as the files may be exposed. In addition, in the case of mobile or wearable devices, the possibility of losing devices is considerably greater because of their small size and mass. Moreover, the greater the number of participating devices, the greater the probability of losing confidentiality.

To prevent these problems, previous studies applied a secret sharing scheme, which is a technique that distributes secrets and shares them with multiple participants. The secret sharing scheme was proposed by Shamir [4] and Blakley [5] independently, and various researchers have developed advanced versions thereof [6][7]. Most of the secret sharing

Received (June 5, 2017), Review Result (August 25, 2017), Accepted (September 10, 2017)

ISSN: 2005-4262 IJGDC

schemes are based on polynomial operations, which require considerable calculation. If such secret sharing schemes are adopted in distributed storage systems with wearable devices, they are subject to excessive operational costs and time consumption. Moreover, because the size of each share is equal to the size of the secret, the communication cost for storing and retrieving files is a further concern for secret sharing employed in distributed storage.

To solve these problems, *Park et al*. proposed a lightweight (k, n)-file sharing scheme [2]. Instead of a polynomial approach, the authors adopted a combinatorial approach, thereby reducing operational and communication costs dramatically. Further, they proposed a scheduling algorithm to minimize the time for file retrieval. However, this scheme does not consider the size of the file share to be stored in each storage. Ignoring the sizes of the file shares can lead to serious efficiency-related problems throughout distributed storage systems, as follows:

- The system should determine the sizes of the file shares to consider the remaining capacity of each storage. Otherwise, there is the possibility that a very large file share may be allocated to a storage with a very low capacity. This can result in rapid space consumption for small capacity storage.
- The system should allocate file shares according to the bandwidth of each storage. If bandwidth is not taken into consideration, a very large file share may be allocated to a low-speed storage, which can greatly increase the time for file retrieval, thereby reducing the efficiency of the system substantially.

Solving the resource consumption problem is highly important because most mobile devices operate on batteries and have limited storage space. Therefore, efficiency problems in distributed storage systems, including mobile devices, must be considered. To solve this problem, we address methods of determining the size of each file share taking into consideration the storage size and bandwidth.

In addition, in [2], the scheme divides the original file into several segments prior to creating the file share. In other words, when there are n storages and at least k are collected, the original file is first divided into $\binom{n}{k-1}$ segments. Given the sizes of the file shares, appropriate sizes for the file segment may not exist, which is an additional important concern.

In this paper, we propose an efficiency-enhanced combinatorial-based file sharing scheme that considers the heterogeneous characteristics of participant wearable devices. Our contributions to the scheme are as follows:

- A file sharing algorithm that can reduce communication problems and overhead while considering the characteristics of the devices, such as the remaining capacity of the storage, CPU speed, and bandwidth.
- An algorithm that determines the size of each segment of the original file to achieve a ratio of file shares.

This study focuses on designing an algorithm that determines the sizes of file shares and provides protocols for storing and retrieving files. This work is an extended version of the short conference paper [1].

The remainder of this paper is organized as follows: First, Section 2 reviews and briefly summarizes the previous related work. Section 3 describes the overall structure of the system in two parts: file sharing and file retrieval. Section 4 describes the two proposed algorithms in detail. Section 5 compares the performance of the proposed scheme with the previous secret sharing scheme through results of the simulation. Section 6 summarizes and concludes this paper in its entirety.

2. Related Work

Ukwandu et al. [8] pointed out various security issues for cloud systems and proposed a secret sharing architecture, RESCUE, for multi-cloud security. The authors noted that the secret sharing scheme is an important technique in encryption. They applied each scheme—Adi Shamir's perfect secret sharing scheme (PSS) [4], Hugo Krawczyk's secret sharing made short or computational secret sharing scheme (CSS) [9], Rabin's information dispersal algorithm (IDA) [10]—to verify the performance of each secret sharing scheme. Moreover, users have to decide whether to use the appropriate secret sharing algorithm based on the type of data.

Marwan et al. [11] proposed a secret sharing-based approach to improve the confidentiality and integrity of medical information. Medical data are stored in cloud storage because they require a large storage system and processing power; however, personal health information is sensitive and must be kept confidential. Therefore, they proposed a framework based on the Shamir secret sharing scheme to ensure the security of medical data stored in the cloud.

Although it is expected that data can be moved to the cloud through wearable devices that contain medical information owing to the development of medical wearable devices, efficiency problems may arise as a result of not considering the capacity and communication speed of various cloud storage systems in [11]. In addition, the framework is not suitable for enhancing security in a cloud environment composed of mobile devices owing to the considerable amount of computation required.

Chirchir et al. [12] proposed a system named SmartSec that uses a secret sharing scheme between several smart devices to enhance privacy. SmartSec focuses on enhancing the privacy of important photos on smart devices. This work aims to improve the security of mobile-based cloud storage; however, the proposed system focuses on protecting only photographs, and is computationally large because it is based on the Shamir secret sharing scheme.

Wang et al. [13] provided a solution with a secret sharing scheme for user media security and a water-marking scheme for authentication. Because mobile devices such as smartphones download or upload media, such as videos or photos, to cloud storage, they cannot trust the media services provided by media cloud service providers. However, this study is not suitable for enhancing the security of various files because the solution is limited to media such as photographs and videos.

A variety of studies [14][15][16], including those listed in this section, attempted to enhance the security of the cloud by applying a secret sharing scheme. However, these studies are unsuitable for enhancing security in a cloud environment with mobile devices. Therefore, in this study, with the focus on increasing security and efficiency in a cloud environment composed of mobile devices, we adopt a lightweight secret sharing scheme and propose a scheduling algorithm for a communication-efficiency and space-efficiency scheme.

3. Overview of the Combinatorial-based File Scheme

Figure 1. Overview of Combinatorial-based File Sharing (a) Storage (b) Retrieval

Figure 1 shows the model of the proposed entire combinatorial-based (k, n)-file sharing scheme. The scheme comprises the processes of file storage and retrieval. In the (k, n)-file sharing scheme, the number of devices constituting the distributed storage system is n and the minimum number of storages capable of recovering the original file stored is k. Figure 1 shows the case where $k = 3$ and $n = 5$.

The participant devices can be configuration devices, PCs, servers, mobile, and wearable devices. Among these, it is expected that the wearable devices will have their own storage, computing power, and communication modules in the near future. In addition, let us assume that all the devices are interconnected via wireless networks, and know the characteristics of other participating devices, such as the remaining storage capacity, computing power, and network speed.

Next, we address the system models for file sharing and retrieval in more detail in the following subsections.

3.1. File Sharing

Figure 1-(a) shows the file sharing process. Let the device that creates a new file and tries to store it to the distributed storages be a **creator**, which is represented as a pair of smart glasses in the figure. When the creator saves the file to the distributed system, it first creates n file shares. For this, the creator conducts the following procedures: the creator determines the sizes of n file shares by considering the heterogeneous characteristics of the participant devices, which are proposed in Section 4. Following this, the creator divides the original file into $\binom{n}{k-1}$ files. Finally, the creator generates n file shares by reconstructing the file segments. Note that each file share is composed of $\binom{n-1}{k-1}$ segments, and that any k among the n file shares include all of the $\binom{n}{k-1}$ files segments, but any $k-1$ among n do not.

After generating the n file shares, the creator saves one of the shares to its own storage, and sends the other $n-1$ file shares to other $n-1$ devices via wireless communication, respectively. To ensure secure and private communication, a secure channel may be employed between the devices.

3.2. File Retrieval

Figure 1-(b) indicates the process of file retrieval from the distributed storage. In this process, let the device that tries to recover the file stored in the distributed storage be a **collector**, which is depicted as a smartphone in the figure. To recover the original file, k file shares must be collected; thus, the collector requests that $k-1$ devices send a file share to the collector. In this study, we do not address methods of selecting the $k-1$ of the $n-1$ devices.

To recover the file, the collector may receive $k-1$ file shares from the $k-1$ devices to collect k file shares. However, duplicated file segments exist in the k shares. Using Figure 1-(b) as an example, the smartphone receives the file shares from the laptop and smart glasses. Because the laptop has segments 1, 4, 5, and the smart glasses have segments 4, 7, 10 in common with the smartphone, receiving the entire file shares from both devices is wasteful. To address this problem, the collector receives the parts of the file shares that include the segments that the collector does not have. Moreover, there is no duplicated content in the downloaded parts of the file shares.

There is a further issue in file retrieval. The ratio of the downloaded parts from the $k-1$ devices should be the same as that of the communication speeds between the collector and the $k-1$ devices. If this is not considered, a device with a lower communication speed may send large parts of its own file share to the collector, which causes a bottleneck.

The download scheduling algorithm determines the part size of each file segment that the collector downloads from the $k-1$ devices. After using the algorithm, the collector sends the results to the $k-1$ devices, and the devices send back the parts of the file sequences to the collector. Finally, the collector retrieves the original file on completion of the transmission.

4. Concrete Processes

4.1. File Sharing Process

The aims of this process are to determine the sizes of the file shares and to generate them. As mentioned earlier, when the sizes of the file shares are determined, the heterogeneous characteristics of the devices, such as the remaining storage capacity and average network speed, are required to prevent inefficient storage usage and bottlenecks. To consider these requirements, the size of each file share may be proportional to the product of the remaining storage size and average

network speed of the device to which the share will be stored. Let $\mathcal{F}$ and F be the original file and its size, respectively, and let $\mathcal{D}_1$, $\mathcal{D}_2$, ⋯, $\mathcal{D}_n$ be the n participant devices. In addition, let $r_1 : r_2 : \cdots : r_n$ be the ratios of the remaining storage sizes of $\mathcal{D}_1$, $\mathcal{D}_2$, ⋯, $\mathcal{D}_n$, and $p_1 : p_2 : \cdots : p_n$ be ratios of their average network speeds. If f_1, f_2, ⋯, f_n are the sizes of the file shares to be allocated to $\mathcal{D}_1$, $\mathcal{D}_2$, ⋯, $\mathcal{D}_n$, respectively, the ideal ratio of the file share is $r_1p_1 : r_2p_2 : \cdots : r_np_n$. According to the Cauchy–Schwartz inequality, the following equation should be maximized:

$$\frac{(f_1r_1p_1 + f_2r_2p_2 + \cdots + f_nr_np_n)^2}{f_1^2 + f_2^2 + \cdots + f_n^2} \tag{1}$$

Eq. (1) is maximized provided $r_1p_1 : r_2p_2 : \cdots : r_np_n = f_1 : f_2 : \cdots : f_n$.

Now, let us discuss the determination of the segment sizes for given sizes of file shares. Let $\mathcal{s}_1$, $\mathcal{s}_2$, ⋯, $\mathcal{s}_\sigma$ be the first, second, ⋯, σ-th file segments, where $\sigma = \binom{n}{k-1}$; let x_j be the size of $\mathcal{s}_j$, which are the unknown variables to be determined. Finally, for $i = 1, 2, \cdots, n$, let $\mathcal{S}_i$ be a set of indexes of file segments that are in the i-th file share. Note that $|\mathcal{S}_i| = \binom{n-1}{k-1}$. Then, the following system of equations can be constructed:

$$\forall i \in \{1,2,\cdots,n\}, \qquad \sum_{j \in \mathcal{S}_i} x_j = f_i \tag{2}$$

Let us consider an $n \times \sigma$ matrix P, where the element in the i-th row and j-th column is one if $\mathcal{s}_j$ is in the i-th file share, or zero otherwise. If $\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_\sigma]^T$ and $\mathbf{f} = [f_1 \quad f_2 \quad \cdots \quad f_n]^T$, Eq. (2) becomes $P\mathbf{x} = \mathbf{f}$. For example, if $n = 4$ and $k = 2$, we construct the following matrix equation:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}$$

The system of equations is composed of $\binom{n}{k-1}$ unknown variables and n equations, where $\binom{n}{k-1} > n$ in most cases; thus, an infinite number of solutions exists. If a tuple of nonnegative unknowns, x_1, x_2, ⋯, x_σ, exists, then there is no problem. This means that the sizes of the file segments can be set as x_1, x_2, ⋯, x_σ. However, the system may not have nonnegative solutions, which is the case where the sizes of file segments cannot be determined.

In turn, the best file share sizes are f_is, where Eq. (1) has a maximum value and Eq. (2) has nonnegative solutions. In addition, because each segment is included in $n - k + 1$ file shares, $f_1 + f_2 + \cdots + f_n = (n - k + 1) \times F$. To conclude this analysis, we derive the following optimization problem:

$$\begin{aligned} \textbf{Maximize}&: \frac{(f_1r_1p_1 + f_2r_2p_2 + \cdots + f_nr_np_n)^2}{f_1^2 + f_2^2 + \cdots + f_n^2} \\ \textbf{subject to}&: f_1 + f_2 + \cdots + f_n = (n - k + 1) \times F \\ &\quad \text{Eq. (2) has nonnegative solutions.} \end{aligned}$$

After f_1, f_2, ⋯, f_n are determined, Eq. (1) is solved and the nonnegative solutions of x_1, x_2, ⋯, x_σ are derived. Now, we create the n file shares. The file shares

creation algorithm for $x_1, x_2, \cdots, x_\sigma$ is given as Algorithm 1.

Algorithm 1. Generation of file shares

Input: $\mathcal{F}, x_1, x_2, \cdots, x_\sigma$
Output: n file shares $\mathcal{f}_1, \mathcal{f}_2, \cdots, \mathcal{f}_n$

1: Divide $\mathcal{F}$ into σ segments $\mathcal{s}_1, \mathcal{s}_2, \cdots, \mathcal{s}_\sigma$ with sizes of $x_1, x_2, \cdots, x_\sigma$, respectively;
2: Initialize $\mathcal{f}_1, \mathcal{f}_2, \cdots, \mathcal{f}_n$;
3: $B \leftarrow \{(b_1, b_2, \cdots, b_n) | b_1 + b_2 + \cdots + b_n = k - 1,\ b_1, b_2, \cdots, b_n \in \{0,1\}\}$;
4: $j \leftarrow 1$;
5: **for all** $(b_1, b_2, \cdots, b_n) \in B$
6: **for** $i = 1$ **to** n
7: **if** b_i is 0
8: $\mathcal{f}_i \leftarrow \mathcal{f}_i || \mathcal{s}_j$;
9: **end if**
10: **end for**
11: $j \leftarrow j + 1$;
12: **end for**
13: **return** $\mathcal{f}_1, \mathcal{f}_2, \cdots, \mathcal{f}_n$;

Algorithm 1 is composed of two stages. The first stage is the division of the original file $\mathcal{F}$ into $\binom{n}{k-1}$ segments; the second stage is the building of the n file shares with the segments. Because Stage 1 is described trivially, we explain the second stage with an example of the case where $n = 4$ and $k = 3$. Then, according to line 3 of the algorithm, set B becomes

$$\{(0,0,1,1), (0,1,0,1), (0,1,1,0), (1,0,0,1), (1,0,1,0), (1,1,0,0)\}.$$

In the iterations in lines 5–12, assume that the order of elements is chosen to be the same as that in the above set. Then, in the first iteration, (b_1, b_2, b_3, b_4) is $(0,0,1,1)$; thus, $\mathcal{f}_1$ and $\mathcal{f}_2$ become $\mathcal{s}_1$, $\mathcal{s}_1$, respectively. Next, (b_1, b_2, b_3, b_4) becomes $(0,1,0,1)$; thus, $\mathcal{f}_1$, $\mathcal{f}_2$, and $\mathcal{f}_3$ become $\mathcal{s}_1 \| \mathcal{s}_2$, $\mathcal{s}_1$, and $\mathcal{s}_2$, respectively. In the same way, after six repetitions in the same manner, the file shares are completely generated as follows:

$$\mathcal{f}_1: \mathcal{s}_1 \| \mathcal{s}_2 \| \mathcal{s}_3, \quad \mathcal{f}_2: \mathcal{s}_1 \| \mathcal{s}_4 \| \mathcal{s}_5, \quad \mathcal{f}_3: \mathcal{s}_2 \| \mathcal{s}_4 \| \mathcal{s}_6, \quad \mathcal{f}_4: \mathcal{s}_3 \| \mathcal{s}_5 \| \mathcal{s}_6.$$

Finally, the generated file shares are returned. Let $\mathcal{D}_c$ be the collector device. Then, of the n shares, $\mathcal{f}_c$ is stored to $\mathcal{D}_c$, and the remaining shares are sent to the other $n - 1$ devices.

4.2. File Retrieval Process

We describe the process for file retrieval. The issue related to the file retrieval is the speed of the network between the collector and the other at least $k - 1$ devices. In this subsection, let $\mathcal{D}_c$ be the collector. The process comprises the procedures for scheduling and recovering. The first procedure determines the number of parts of file segments downloaded from each device. Following this, the collector requests other devices to send parts of the file segments according to the results, and receives them. Finally, the original file is retrieved in the second procedure.

Let us assume that the devices that the collector requests are $\mathcal{D}_{q_1}, \mathcal{D}_{q_2}, \cdots, \mathcal{D}_{q_{k-1}}$, where $1 \le q_1 < q_2 < \cdots < q_{k-1} \le n$ and $c \notin \{q_1, q_2, \cdots, q_{k-1}\}$. Let the ratio of the

communication speed between $\mathcal{D}_c$ and $\mathcal{D}_{q_1}$, $\mathcal{D}_{q_2}$, ⋯, $\mathcal{D}_{q_{k-1}}$ be $e_{q_1}:e_{q_2}:\cdots:e_{q_{k-1}}$. In addition, consider an $n \times \sigma$ matrix ST (scheduling table)), where the element in the i-th row and the j-th column, $ST_{(i,j)}$, is the number of parts of the j-th segment $\mathcal{s}_j$ that $\mathcal{D}_c$ receives from $\mathcal{D}_j$. The goals of the scheduling procedure are as follows:

1) For $j = 1, 2, \cdots, \sigma$: if $j \notin \mathcal{S}_c$, the sum of elements in the j-th column is $|\mathcal{s}_j|$; otherwise, it is zero.
2) Let $ST_i = ST_{(i,1)} + ST_{(i,2)} + \cdots + ST_{(i,\sigma)}$. Then, $ST_{q_1}:ST_{q_2}:\cdots:ST_{q_{k-1}}$ is similar to $e_{q_1}:e_{q_2}:\cdots:e_{q_{k-1}}$ at best.

In addition, the amount of data that $\mathcal{D}_c$ should download from the $k-1$ devices is $F - f_c$. Therefore, if for $i = 1, 2, \cdots, k-1$, $z_{q_i} = \frac{e_{q_i} \times (F - f_c)}{e_{q_1} + e_{q_2} + \cdots + e_{q_{k-1}}}$; then, z_{q_i} is the objective of the total amount of data transmitted by $\mathcal{D}_i$. Let PT be an $n \times \sigma$ matrix, where the element in the i-th row and the j-th column, $PT_{(i,j)}$, is $|\mathcal{s}_j|$ if $\mathcal{s}_j \in \mathcal{S}_c$, or zero.

Algorithm 2. Scheduling the amount of downloaded data

Input: $q_1, q_2, \cdots, q_{k-1}, z_{q_1}, z_{q_2}, \cdots, z_{q_{k-1}}$
Output: ST

Initialize ST as a zero $n \times \sigma$ matrix;
$W \leftarrow \{1,2,\cdots,\sigma\} \backslash \mathcal{S}_c$;
for $i = 1$ to $k-1$ **do**
    for all $j \in \mathcal{S}_{q_i} \cap W$ **do**
        $ST_{(q_i,j)} \leftarrow PT_{(q_i,j)}$;
    end for
    $W \leftarrow W \backslash \mathcal{S}_{q_i}$;
end for
Reorder $(q_1, q_2, \cdots, q_{k-1})$ to $(u_1, u_2, \cdots, u_{k-1})$ that satisfies:
    $ST_{u_1} - z_{u_1} \geq ST_{u_2} - z_{u_2} \geq \cdots \geq ST_{u_{k-1}} - z_{u_{k-1}}$;
$tmp \leftarrow ST_{u_1} - z_{u_1}$;
for $\ell = k-1$ to 1 **do**
    while $ST_{u_1} > z_{u_1}$ **do**
        $A \leftarrow \{j | ST_{(u_1,j)} \neq 0\}$;
        Reorder the elements of $A \cap S_{q_\ell}$ to $(\pi_1, \pi_2, \cdots, \pi_\mu)$ that satisfies $P(\mathcal{s}_{\pi_1}) \leq P(\mathcal{s}_{\pi_2}) \leq \cdots \leq P(\mathcal{s}_{\pi_\mu})$, where $\mu = |A \cap S_{q_\ell}|$.
        for $c = 1$ to μ **do**
            $v \leftarrow \min\{ST_{(u_1,\pi_c)}, ST_{u_1} - z_{u_1}, z_{u_\ell} - ST_{u_\ell}\}$;
            $ST_{(u_1,\pi_c)} \leftarrow ST_{(u_1,\pi_c)} - v$;
            $ST_{(u_\ell,\pi_c)} \leftarrow ST_{(u_\ell,\pi_c)} + v$;
        end for
    end while
end for

22: **If** tmp and $\sum_{j=1}^{\sigma} ST_{(u_1,j)} - r_{u_1}$ are the same
23: **return** ST;
24: **end if**
25: **goto** 9;

Algorithm 2 is the aforementioned download scheduling algorithm. Before describing the algorithm, we introduce a popularity function $P: \{\mathcal{s}_1, \mathcal{s}_2, \cdots, \mathcal{s}_\sigma\} \to \mathbb{Z}$, where $\mathbb{Z}$ is a set of all the integers, as follows:

$$P(\mathcal{s}_t) = \sum_{i=1}^{k'} \delta(\mathcal{s}_t, q_i), \text{where } \delta(\mathcal{s}_t, q_i) = \begin{cases} r_{q_i} - ST_{q_i} & \text{for } r_{q_i} > ST_{q_i} \text{ and } t \in \mathcal{S}_{q_i} \\ 0 & \text{otherwise} \end{cases}$$

Now, we describe this algorithm. Lines 1–8 represent a procedure for initializing the matrix ST greedily, and lines 9–25 represent a procedure for adjusting ST to satisfy the abovementioned goals. After Line 9, $\mathcal{D}_{u_1}$ and $\mathcal{D}_{u_{k-1}}$ are the most over-allocated and under-allocated devices, respectively. Therefore, in lines 10–25, the allocated amount moves from $\mathcal{D}_{u_1}$ to the under-allocated devices.

In terms of lines 10–25, the elements of $A \cap S_{q_\ell}$ are indexes of the candidates of segments whose parts will be sent from $\mathcal{D}_{u_1}$ to $\mathcal{D}_{u_\ell}$. In order of $\mathcal{s}_{\pi_1}$, $\mathcal{s}_{\pi_2}$, ⋯, $\mathcal{s}_{\pi_\mu}$, they move from $\mathcal{D}_{u_1}$ to $\mathcal{D}_{u_\ell}$ until $ST_{u_i} = r_{u_1}$ or $ST_{u_\ell} = r_{u_\ell}$.

The iteration in lines 9–25 proceeds until ST_{u_1} and r_{u_1} are the same or ST_{u_1} remains constant during one round of iteration. The reason for the first is that the two values are the same; that is, ST satisfies the goal. The reason for the second is that ST will no longer be optimized.

Moreover, although this algorithm comprises some iterations, it must be terminated. This is because $ST_{u_1} - r_{u_1}$ decreases for every round of the iteration; however, the equation yields a nonnegative value. Therefore, the equation value becomes zero or stops decreasing when both correspond to the termination conditions.

After conducting the scheduling algorithm, the collector requests k' devices to send the parts of file segments. When the collector receives all the parts, he or she recovers the original file using these parts.

5. Simulation Results

In this section, we analyze the performance of the proposed file sharing scheme based on that of previous file sharing scheme. We compared Shamir's secret sharing scheme with the proposed scheme among previous file sharing schemes. This is because Shamir's scheme is the most popular file sharing scheme and widely used for distributed storage. These simulations are the results of experiment on Macintosh OS, 8GB RAM, and 2.8 GHz Intel Core i5 environment.

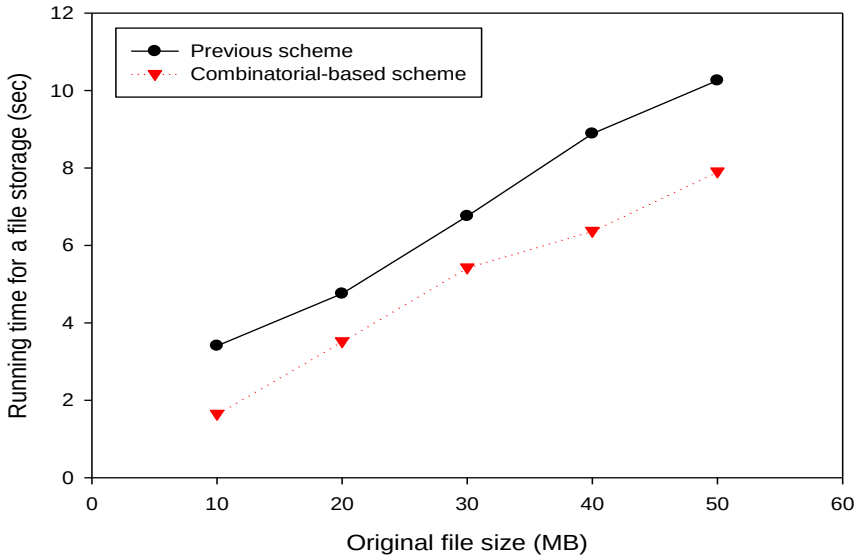


Figure 2. The File Storage Time of (3,5)-File Sharing

Figure 2 shows a graph of operation times for file storage for the case where $k = 3$ and $n = 5$. The black solid and red dotted lines represent the time at which previous scheme and our proposed scheme are adopted, respectively. The capacity and bandwidth of each device are considered to simulate our scheme. The operation times for both the proposed and previous schemes are shown to be almost proportional to the size of the original file. Moreover, the proposed scheme requires less time to store a file to the distributed storage system than previous scheme.

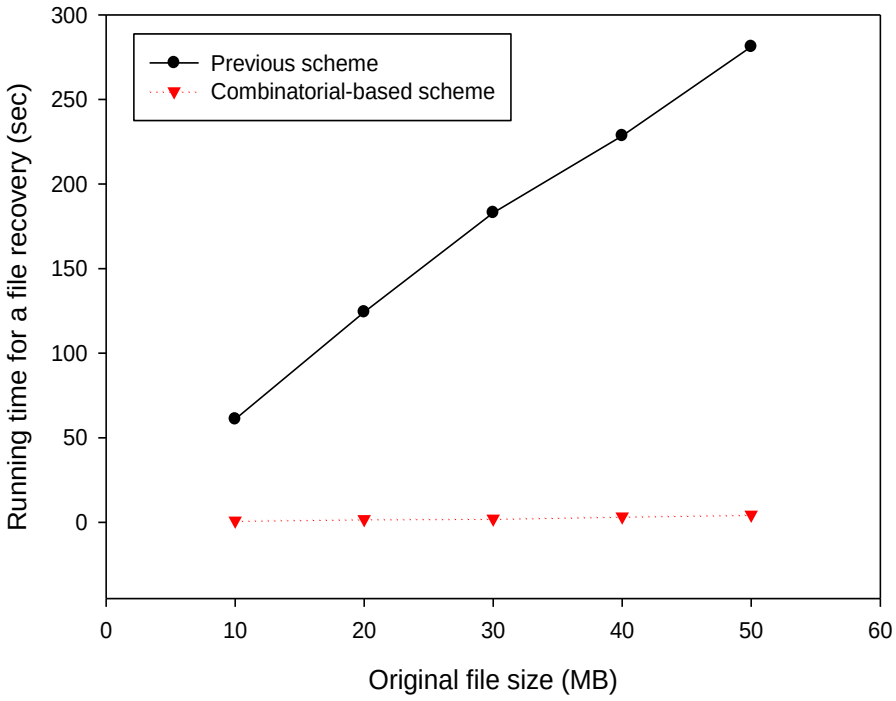


Figure 3. The File Recovery Time of (3,5)-File Sharing

Figure 3 shows the operation times for file retrieval when the combinatorial-based (3,5)-file sharing scheme and existing (3,5)-secret sharing scheme are used for the distributed storage. As shown in the graph, during this process, the proposed scheme exhibits a large performance difference in the file recovery process. The graph of the operation time for previous secret sharing, which is represented by the black solid line, restores the file using the polynomial method; thus, the operation time for file retrieval is considerably larger than in the case of our combinatorial-based file sharing, which is shown by the red dotted line. The previous scheme's process requires a duration of more than 1 min (60.96 s) for a file of 10 MB, and more than 4 min (281.15 s) to recover a 50 MB file. However, the combinatorial-based process requires approximately 1 s to receive a 1 MB file, and approximately 4.5 s to recover a 50 MB file. The combinatorial-based file retrieving scheme performs approximately 65 times faster than previous scheme on average.

As shown in these two figures, the combinatorial-based file sharing indicates that the performance of file storage and retrieval increases more than that of the existing file sharing scheme. In particular, it greatly reduces the time required for file recovery. Unlike the conventional scheme, which has a high computational complexity, its computational cost has been proven to be reduced considerably. Moreover, it is confirmed that it can be used in mobile devices with considerably less computing power than that of ordinary computers.

6. Conclusion and Future Work

In this paper, we noted that it is important to solve the efficiency problem because various mobile devices can constitute distributed storage systems and these operate as a battery. Thus, we proposed a combinatorial-based file sharing scheme for a distributed storage system with heterogeneous personal devices. Because this scheme considers the characteristics of each device, it can use the distributed storage device more efficiently and the original file can be stored and retrieved more quickly. The comparison of the simulation results of the existing and proposed file sharing schemes proved that efficiency is improved considerably by reducing the time required for file storage and retrieval. However, the simulation in this study compared only the time spent on file storage and recovery operations, and the communication speed of each device was not considered. Therefore, the goal of future work is to derive the results considering not only computation time but also various characteristics of each device.

Acknowledgments

This work was partly supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. 2017-0-01705, Development of SDN Platforms and Security Schemes for Service Scalable and Trustable IoT Environments) and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2017R1C1B5018116).

This paper is a revised and expanded version of a paper entitled Efficient Scheme for Generating File Shares in Combinatorial-based File Sharing with Distributed Cloud Storage presented at the International Conference on Green and Human Information Technology 2017 (ICGHIT 2017), Hangzhou, China, February 15-17, 2017.

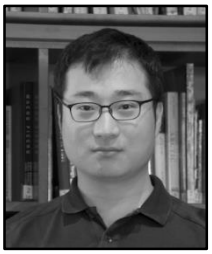
References

[1] J. E. Park, B. Bold and Y. H. Park, "Efficient Scheme for Generating File Shares in Combinatorial-based File Sharing with Distributed Cloud Storage", International Conference on Green and Human Information Technology (ICGHIT), **(2017)**; Hangzhou, China.

[2] Y. H. Park, E. D. Lee and S. W. Seo, "Lightweight (k, n)-file Sharing Scheme for Distributed Storages with Diverse Communication Capacities", IEEE Conference on Communications and Network Security, **(2014)**; San Francisco, CA.
[3] N. Paunkoska, V. Kafedziski and N. Marina, "Improved Perfect Secrecy of Distributed Storage Systems Using Interference Alignment", 8th International Congress on Ultra-Modern Telecommunications and Control Systems and Workshops (ICUMT), **(2016)**; Lisbon, Portugal.
[4] A. Shamir, "How to Share a Secret", Communications of the ACM, vol. 22, no. 11, **(1979)**, pp. 612-613.
[5] G. R. Blakley and G. A. Kabatianskii, "Linear Algebra Approach to Secret Sharing Schemes", Error Control, Cryptology, and Speech Compression, Springer, **(1994)**, pp. 33-40.
[6] N. Tentu, B. Mahapatra, V. C. Venkaiah and V. K. Prasad, "New Secret Sharing Scheme for Multipartite Access Structures with Threshold Changeability", International Conference on Advances in Computing, Communications and Informatics (ICACCI), **(2015)**; Kochi.
[7] S. Chen, Y. Chen, H. Jiang, L. T. Yang and K. C. Li, "A Secure Distributed File System Based on Revised Blakley's Secret Sharing Scheme", IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, **(2012)**; Liverpool.
[8] E. Ukwandu, W. J. Buchanan, L. Fan, G. Russell and O. Lo, "RESCUE: Resilient Secret Sharing Cloud-Based Architecture", IEEE Trustcom/BigDataSE/ISPA, **(2015)**; Helsinki.
[9] H. Krawczyk, "Secret Sharing Made Short", Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO), Springer-Verlag, **(1993)**.
[10] M. O. Rabin, "Efficient Dispersal of Information for Security Load Balancing and Fault Tolerance", Journal of the ACM, vol. 36, no. 2, **(1989)**, pp. 335-348.
[11] M. Marwan, A. Kartit and H. Ouahmane, "Secure Cloud-based Medical Image Storage Using Secret Share Scheme", 5th International Conference on Multimedia Computing and Systems (ICMCS), **(2016)**; Marrakech.
[12] B. Ben Chirchir, X. Zhang, M. Li, Q. Qian, N. Ruan and H. Zhu, "SmartSec: Secret Sharing-based Location-aware Privacy Enhancement in Smart Devices", IEEE/CIC International Conference on Communications in China (ICCC), **(2015)**; Shenzhen.
[13] H. Wang, S. Wu, M. Chen and W. Wang, "Security Protection between Users and the Mobile Media Cloud", IEEE Communications Magazine, vol. 52, no. 3, **(2014)**, pp. 73-79.
[14] C. Padsala, R. Palav, P. Shah and S. Sonawane, "Survey of Cloud Security Techniques", International Journal for Research in Applied Science & Engineering Technology, vol. 3, no. 3, **(2015)**, pp. 47-50.
[15] J. V. Bharambe and R. K. Makhijani, "Secured Data Storage and Retrieval in Multi-Cloud using Shamir's Secret Sharing Algorithm", International Journal of Computer Science and Engineering, vol. 2, no. 3, **(2013)**, pp. 15-19.
[16] M. Padmavathi, D. Sirisha and A. Lakshman Rao, "The Security of Cloud Computing System Enabled by Shamir's Secret Sharing Algorithm", International Journal of Research Studies in Science Engineering and Technology, vol. 1, no. 9, **(2014)**, pp. 103-109.

Authors

Jung-Eun Park, she received B.S. degree from Sookmyung Women's University, Seoul, South Korea, in 2017. Her research interests include network security and secure storage in cloud computing. She is currently a M.S. student affiliated with Division of Computer Science of Sookmyung Women's University, Seoul, South Korea.



Young-Hoon Park, he received B.S., M.S., and Ph.D. degrees from Seoul National University, Seoul, South Korea, in 2006, 2008, 2013, respectively. He was a senior engineer with Cloud Computing Lab, Software Center, Samsung Electronics, Suwon, South Korea. He was a Postdoctoral Researcher sponsored by Brain Korea 21 with School of Electronic Engineering, Seoul National University. He is currently an Assistant Professor with Division of Computer Science, Sookmyung Women's University,

Seoul, South Korea. He is also a committee member of the Institute of Electronics and Information Engineers, and International Conference on Green and Human Information Technology (ICGHIT) 2018. His research area includes network security, cryptography, and system optimization.

