IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 11, 2019, accepted November 25, 2019, date of publication November 28, 2019, date of current version December 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2956681

High-Speed Searching Target Data Traces Based on Statistical Sampling for Digital Forensics

DOOWON JEONG AND SANGJIN LEE
Digital Forensics Research Center, Korea University, Seoul 02841, South Korea
Corresponding author: Sangjin Lee (sangjin@korea.ac.kr)

This work was supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korea Government (MSIT) (No. 2018-0-01000, Development of Digital Forensic Integration Platform).

ABSTRACT As technology of manufacturing storage medium advances, data storage capacity has been increasing exponentially. This pervasiveness has made a forensic examination time-consuming and difficult. If a file system of data storage remains intact, an examiner can find files that would be important evidence by analyzing hierarchy, name, time information, etc. of files and folders. However, as anti-forensic techniques such as metadata destruction and disk format are widely known, the data search based on the file system becomes more impractical. Besides, significant evidences could be stored in the unallocated area; investigating the entire area of data storage is still important. The famous methods of exploring the existence of evidence are hash comparison and random sampling. The hash comparison that calculates hash for all sectors and compares them can detect all fragments of the evidence. However, it requires an enormous amount of time and computing resources. Whereas the random sampling takes much less time as it exploits a portion of data storage, but it involves the risk of false-negative; this fact is critical to forensic examiners. In this paper, we blend the merits of both methods to make false-negative zero and to reduce the processing time extremely at the same time. We use 16-byte values in a sector instead of traditional hash to filter out the unmatched sector. The values are statistically selected based on the frequency of occurrence according to offset. The effectiveness of our methodology is evaluated through several experiments.

INDEX TERMS Forensics, computer crime, security, data acquisition.

I. INTRODUCTION
Among various forensic techniques, data search is the most basic method and is is one of the most significant techniques. In digital forensic field, the search means that a forensic examiner finds out already known data or file from a plaintiff or other case in target machine such as PC, laptop, smartphone, USB, database, etc. Especially in e-discovery or leaking confidential information case, whether the known file exists in the suspect's system can be a very important key [1], [2].

The simplest method for searching is matching every bit of target data and data storage; this method is called bytewise matching [3]. The bytewise matching can find the target files in any data storage, however, it is time-consuming and it requires considerable resources of a computer. Besides, time for the operation is hard to estimate as it is greatly affected by a specification of the analysis system and performance of the storage medium. Analyzing a file system is an another method to identify the known data [4]. By using this method, files would be sorted by time or name; the examiner can find the data created at a specific time or the peculiar word relative to a case. However, this method can produce the desired result only when the metadata area remains intact. If the examiner is unable to interpret the metadata or target file is deleted, this method can not be applied to forensic investigation. The keyword search and indexing technique can also be used. To extract words or sentences from a file, an automated program is required to parse an internal structure of the file and create a database for searching [5]. Based on parsed data that can be perceived by humans, a technique to find similar photos or sentences is also applicable to solve the search problem [6], [7]. Image processing, plagiarism detection, and authorship verification are sub-fields of the methodology. This technique must support a high degree of human intuition and judgment locating the subtle variation.

Although the state-of-the-art techniques have been released, the concept of bytewise matching is still important to forensic examiners; it guarantees 100 percent true positive and 0 percent false negative. The filtering method using file system is vulnerable even to tiny anti-forensic techniques such as deleting file, formatting, renaming the file, changing

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Imran Tariq.

time information of metadata, etc. The keyword search and indexing have an inherent risk of false-negative as the analysis of file system and file format must take precedence. The false-negative can be increased when the target files are formatted by the customized versions. Although AI techniques have been researched to identify the similar graphics and text, they can not ensure 0 false-negative rates (FNR). The fact that FNR would be non-zero is a sensitive issue especially in digital forensics since a piece of critical evidence could be missed permanently.

The most famous method of bytewise matching is the complete enumeration that compares hash values in a sector between a storage device and a target file. The hash comparison method always makes FNR zero and false positive rate (FPR) near zero. However, its complexity gets impractical as the size of the target increases [8]. It's no wonder that ordinary people may generate gigabytes of information a month. As smartphones that have terabytes memory have also been released, the size of data that needs to be analyzed has been increasing exponentially. The increase in storage capacity has presented the extreme challenges for searching the target data due to the requirement of considerable resource and time [9], [10]. As searching data has become resource-intensive and time-consuming, examination cost has also increased significantly as well [11]. This issue can lead to a more serious problem when the examiner investigates not only a person but a corporation or organization group that has a huge number of computer systems. To overcome the limitation of hash comparison, random sampling was suggested based on probability statistics [11], [12], [13]. The random sampling drastically reduces usage of memory and time, however, it can not assure zero FNR; there is a trade-off between time and accuracy.

This paper proposes a novel methodology to search the known target data on the data storage. The main concept of our approach is merging characteristics of complete enumeration and random sampling described above. Note, the objective of our approach focuses on not semantically identical data but bytewise matching data. We do not use traditional cryptographic hashing but 16-byte values statistically chosen in a sector. Our method, which adapts probability and statistics, is as follows: (1) select 16-byte values for all sectors of the target data by statistical sampling, (2) explore sectors on the data storage by region unit, (3) compare chosen 16-byte values in a sector between a storage device and a target data. The region means that the minimum amount of data that can be read by the file system. We verify that FPR gets near zero as the collision resistance of statistically selected 16-byte values is incredibly similar to the MD5 hash algorithm. The main contributions of this work can be summarized as follows.

- We propose a novel methodology that takes advantage of complete enumeration and random sampling. Through this approach, the forensic examiners can have their cake and eat it; computing resources and processing time.
- We address the probability distribution for offset-value of data storage. The distribution may be used in various forensic areas such as data carving, triage, abnormality detection, etc.
- We verify that only 16-byte values in a sector can be forensic signature if they are selected by statistically. In the forensic search problem, this method can be used as an alternative to the cryptographic hash.

The remainder of this paper is organized as follows: Related works are presented in Section II. In Section III, our first proposed method named random offset sampling is described. Next, we introduce our advanced method in Section IV. In Section V, we prove that the introduced method is useful in practice through an experiment. Finally, we give the conclusion and the future work in Section VI.

II. RELATED WORK

Various forensic methods for detecting target data have been researched. The previous studies acknowledged that since the full investigation such as complete enumeration search is impractical, they focused on making FNR acceptable to the forensic examiner. To develop the methods more significantly and practically, the previous researches tried to sharpen discrimination algorithms and to reduce the size of data storage.

A. DISCRIMINATING BETWEEN DATA

A comparison bit by bit can verify whether two data are identical or not. However, the method is considered impractical as it requires comprehensive computing resources. Therefore, forensic examiners have used hash algorithms like MD5 and SHA256. The hash is appropriate to find the same data since the hash functions generate very different hash value for a similar bitstream. To identify similar data, bytewise approximate matching algorithms were developed; ssdeep [14], sdhash [15], and mrsh-v2 [16]. They show notable achievement but it takes more time than cryptographic hash. For this reason, the forensic examiners tend to use the algorithms when they analyze not disk volume but a limited number of files. To reduce processing time and memory usage rate, Penrose *et al.* [17] used a bloom filter of target files. They generated the entries for more than 200 million blocks in bloom filter and then verified that fragments of target data on storage media can be detected in 2 hours even though using legacy equipment. McKeown *et al.* [18] suggested first 4KB and last 4KB of the file as forensic signatures. They verified that the hash value for the part of the file can be used for discriminating whether two files are identical with high low FPR. Although it focused on using filesystem data to find the files, unlike previous researches, only part of a file can be used as an input of hash functions in digital forensics. Note, Penrose *et al.* [17] and McKeown *et al.* [18] designed their algorithm made for a faster initial scan. The methods focus on hitting at least one sector of target data in a short time.

B. REDUCING THE QUANTITIES OF STORAGE

The filtering techniques based on contents of a file have been studied as an alternative to the full investigation. Da Cruz Nassif and Hruschka [19] downsized target medium

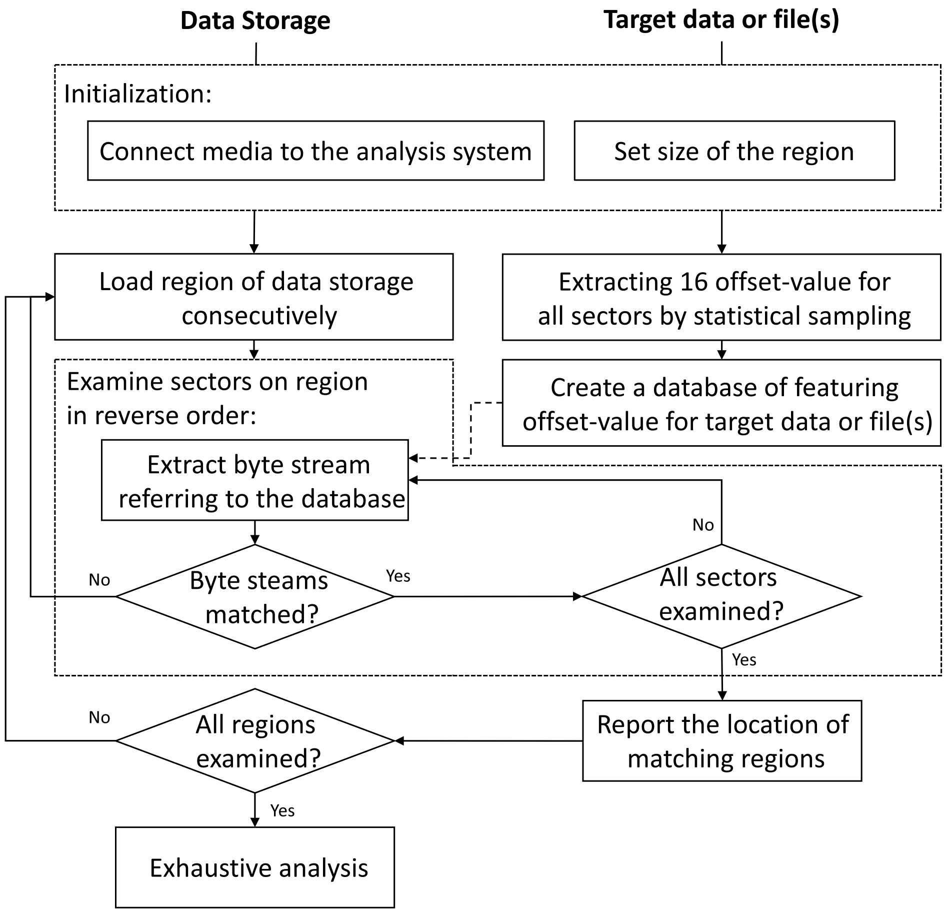

Data Storage
Target data or file(s)
Initialization:
Connect media to the analysis system
Set size of the region
Load region of data storage consecutively
Extracting 16 offset-value for all sectors by statistical sampling
Examine sectors on region in reverse order:
Create a database of featuring offset-value for target data or file(s)
Extract byte stream referring to the database
No
Byte steams matched?
Yes
All sectors examined?
No
Yes
All regions examined?
No
Report the location of matching regions
Yes
Exhaustive analysis

FIGURE 1. The flow chart of the proposed algorithm.

by document clustering and Beebe *et al.* [20] classified files through n-gram vector and data type. Iqbal *et al.* [21] also suggested an efficient resizing method based on the e-mail file format. For example, the messages sent and received are filtered by time information or relationship with people. Decherchi *et al.* [22] and Dagher and Fung [23] focused on not file type but contents itself as a standard of filtering. Beebe *et al.* [24] and SaiKrishna *et al.* [25] parsed text by analyzing the internal structure of the file first and then searched the data including specific keywords. The keywords could be generated by a predefined process or defined by a forensic analyst. Image file clustering also has been researched to save the examiner's trouble. Caldelli *et al.* [26] proposed a function that supports clustering given images to reduce computational complexity and to improve the true-positive rate (TPR). Amerini *et al.* [27] proposed a method for camera identification through image clustering. Villalba *et al.* [28] targeted images taken by smartphones. Based on a characteristic of the images such as sensor pattern noise, they proposed a clustering method using the combination of hierarchy and the use of sensor noise. Lin and Li [29] introduced the challenges of large-scale image clustering and proposed a novel framework based on camera fingerprints without a training process. The described methods can be used when the contents of the file are already extracted from data volume. The preprocess phase such as extracting files from volume and interpreting raw data of the files would be impractical as it can be so time-consuming and resource-intensive on the field. Another drawback is that the methods can not support finding data in the unallocated area when the examiner analyzes data storage.

Reducing the size of target object itself can be a good alternative when the forensic examiner collects evidence. In the digital forensic community, it is called a 'triage' [30]. Triage means that an onsite or field approach for providing the identification, analysis, and interpretation of digital data without requirement of having to take the systems back to the lab for an in-depth examination or acquiring a complete forensic image. Real-time triage [31], digital media triage [32], and case-based reasoning method [33] were proposed as sub-field of triage, however, the studies are applicable to only live system. Quick and Choo [34] developed a process reducing data volume when imaging. They focused on selecting key files such as operating system artifacts, messages, logs, etc. These triage methods are rule-based search algorithm so it could handle the large volume of data effectively.

Nevertheless, the suggested methods have a critical limitation that they are vulnerable to anti-forensic attack. They could be inapplicable when the file system or database is not interpreted correctly. For instance, if data storage is formatted by a customized file system, triage methods are inapplicable as the file system can not be interpreted and the files on the volume can not be identified. It is also possible to miss the crucial evidence because indexing or data carving is limited in the live system.

Comparing hashes in sectors between data storage and target file would be the most reasonable method if an examiner should find already known file or data completely [8]. When the forensic examiner must gain knowledge about the existence of target file or data such as child pornography, confidential document, fragment of malware, etc. in data storage damaged by anti-forensic technique, comparison of raw data could be the only way. The simplest and the most accurate method is the complete enumeration for raw data, however, as mentioned in Section I, it has become an impractical method as the storage capacity has increased. Garfinkel *et al.* [35] proposed random sampling for rapid triage. Instead of comparing all data in the data storage, they compared hash between the target data and samples randomly chosen from data storage. The concept of this method can be pragmatic in that it explores raw data without considering the file system or file format.

1) RANDOM SAMPLING

Random sampling for data storage is correlated to hypergeometric distribution; a kind of an urn problem in probability theory, where an urn consists of two colors ball. In random sampling [35], two balls mean null and non-null sectors. Then the probability of not finding non-null sectors can be presented as follow:

$$P(X = x) = h(x; m, K, N) = \frac{\binom{K}{x}\binom{N-K}{m-x}}{\binom{N}{m}} \tag{1}$$

N, K, m, and x respectively represent a total number of sectors, total non-null sectors count, considered the quantity of samples, and the number of successful attempts to find non-null sectors. Although the random sampling proposed to distinguish between the null sector and the non-null sector, it carried the meaningful footstep as it was presented as an alternative to complete enumeration. Taguchi [12] and Canceill [13] suggested the methodology to identify a key file employing the random sampling that searches the volume by the transaction block which is the logical size of the volume. Transaction block is generally 4 KB and it corresponds to the cluster concept. As the search and comparison work by transaction block requires less I/O operation than by sector unit, this approach was able to improve the processing speed.

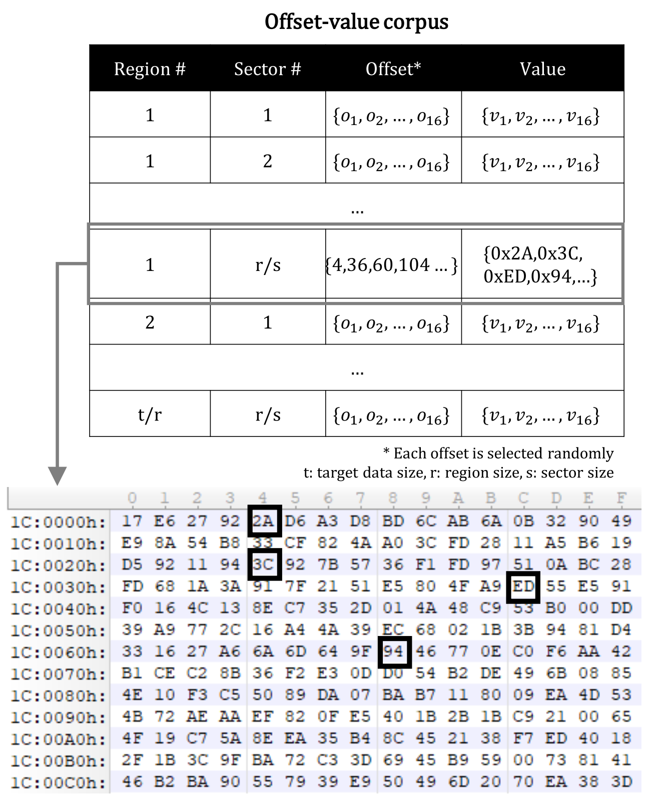
Even though random sampling is an effective method, a lot of time must be still required when working on the large-scale volume especially. Specifically, the hard disk may cause a considerable pressure to the computer system as I/O processing speed is much slower than CPU and memory. To solve this problem, Bharadwaj and Singh [11] suggested a novel approach that divides the storage media into the region and explores in consecutive order. The region consists of sectors and the size of the region is determined experimentally considering I/O operation. Samples are chosen randomly in each region and then compared with target data by the hash value. They verified that this approach can assist in alleviating the number of read requests and concerned time complexities.

Although the random sampling provides high-speed matches, it still has two drawbacks: the inapplicability in detecting whole data and the existence of false-negative. Though this method shows good performance when finding a piece of target data, it cannot be used to find all sectors of target data. If the target data are fragmented in volume, there is no choice to solve the problem. The existence of false-negative is a more critical factor in forensic investigation. As Hirano *et al.* [36] evaluated the random sample's effectiveness experimentally, the method cannot find files less than 720 bytes on Windows OS. The evidence may never be found anymore if the evidence cannot be detected although the target was in the storage media. As the random sampling was based on the assumption that the data of the file is uniformly saved in each region, it sometimes does not properly work in the field. To remedy the drawbacks, we propose a novel method based on the ideas of random sampling and complete enumeration. Our method that is going to be described can balance the requirements of digital forensics: time efficiency and evidence detection with zero FNR.

III. PROPOSED METHOD 1 : RANDOM OFFSET SAMPLING

In our firstly proposed method, we use the byte values randomly chosen in a sector as a forensic signature. In this method, it is assumed that the probability distribution of byte values for each offset is a uniform distribution. We choose 16-byte values arbitrarily in a sector as the collision resistance gets to be $1/2^{128}$ on the assumption. Though using more bytes will improve accuracy, we considered that using more byte stream will have a negative effect on time performance as it requires more comparison operations. In addition, the 16-byte stream's accuracy is enough to find target data through the experiment described in Section V; therefore, similar to MD5, we used 16-byte values as the signature.

First, the 16 offsets randomly selected for each sector of the target file, the corresponding byte values are stored with offset in the offset-value corpus. After then, sectors of data storage are explored by the region unit. This method explores all sectors by region unit to reduce the delay time due to frequent I/O; this idea comes from [11]. The default size of the region is set as the cluster size, however, memory tolerance should be considered in the analysis environment system. This helps to reduce processing time especially analyzing HDD. When exploiting data storage, comparison sectors on region are examined in reverse order. For example, when the region size is 4KiB, we firstly compare between all nth sectors that are $n \bmod 8 (= 4\text{KiB}/512\text{Byte}) \equiv 0$ and the last sector



Offset-value corpus

Region #	Sector #	Offset*	Value
1	1	$\{o_1, o_2, \dots, o_{16}\}$	$\{v_1, v_2, \dots, v_{16}\}$
1	2	$\{o_1, o_2, \dots, o_{16}\}$	$\{v_1, v_2, \dots, v_{16}\}$
…			
1	r/s	{4,36,60,104 …}	{0x2A,0x3C, 0xED,0x94,…}
2	1	$\{o_1, o_2, \dots, o_{16}\}$	$\{v_1, v_2, \dots, v_{16}\}$
…			
t/r	r/s	$\{o_1, o_2, \dots, o_{16}\}$	$\{v_1, v_2, \dots, v_{16}\}$

* Each offset is selected randomly
t: target data size, r: region size, s: sector size

```
            0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
1C:0000h:  17 E6 27 92 2A D6 A3 D8 BD 6C AB 6A 0B 32 90 49
1C:0010h:  E9 8A 54 B8 33 CF 82 4A A0 3C FD 28 11 A5 B6 19
1C:0020h:  D5 92 11 94 3C 92 7B 57 36 F1 FD 97 51 0A BC 28
1C:0030h:  FD 68 1A 3A 91 7F 21 51 E5 80 4F A9 ED 55 E5 91
1C:0040h:  F0 16 4C 13 8E C7 35 2D 01 4A 48 C9 53 B0 00 DD
1C:0050h:  39 A9 77 2C 16 A4 4A 39 EC 68 02 1B 3B 94 81 D4
1C:0060h:  33 16 27 A6 6A 6D 64 9F 94 46 77 0E C0 F6 AA 42
1C:0070h:  B1 CE C2 8B 36 F2 E3 0D D0 54 B2 DE 49 6B 08 85
1C:0080h:  4E 10 F3 C5 50 89 DA 07 BA B7 11 80 09 EA 4D 53
1C:0090h:  4B 72 AE AA EF 82 0F E5 40 1B 2B 1B C9 21 00 65
1C:00A0h:  4F 19 C7 5A 8E EA 35 B4 8C 45 21 38 F7 ED 40 18
1C:00B0h:  2F 1B 3C 9F BA 72 C3 3D 69 45 B9 59 00 73 81 41
1C:00C0h:  46 B2 BA 90 55 79 39 E9 50 49 6D 20 70 EA 38 3D
```

FIGURE 2. Proposed method 1: Example of offset value corpus.

Algorithm 1 Random Offset Sampling

```
1: Inputs:
       Data storage, Target data
2: Initialize:
       s - Sector size, r - Region size,
       d - Data storage size, t - Target data size
       L - List of matching regions
3: for $i = 1$ to $d/r$ do
4:     $R^D{}_i = \{S^D{}_1, S^D{}_2, \dots, S^D{}_{r/s}\}$
5:     for $j = 1$ to $t/r$ do
6:         $R^T{}_j = \{S^T{}_1, S^T{}_2, \dots, S^T{}_{r/s}\}$
7:         for $k = 1$ to $r/s$ do
8:             $V^D \leftarrow S^D{}_k \times O_{j,k}$
9:             $V^T \leftarrow S^T{}_k \times O_{j,k}$
10:            for $l = 1$ to 16 do
11:                if $V^D{}_l \neq V^T{}_l$ then
12:                    goto line 5
13:                end if
14:                if $l = 16$ then
15:                    $L \leftarrow (i, j, k)$
16:                end if
17:            end for
18:        end for
19:    end for
20: end for
```

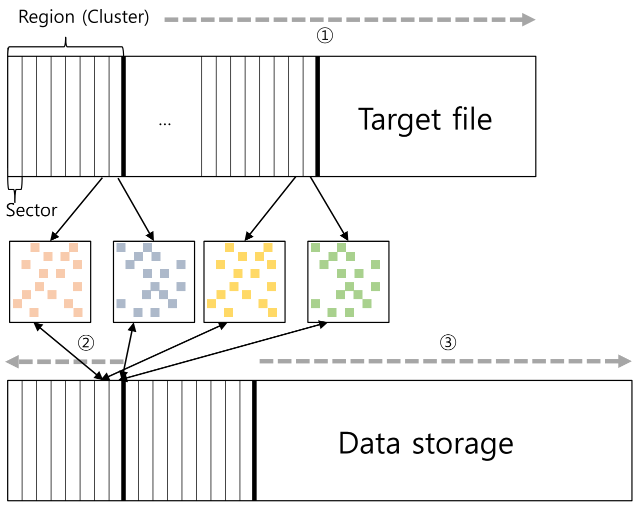


FIGURE 3. The search sequence of random offset sampling.

on the regions. If they are matched, we continue exploring the preceding sectors; if not, the next region is examined. This process progresses consecutively until the last region (See Figure 3). Algorithm 1 shows the process of random offset sampling. $R^D{}_i$ is the ith region of data storage and $R^T{}_j$ is the jth region of target data. In the line 4 and line 6, S_n means an nth sector of the region R. In the line 8, $O_{j,k}$ means an offset set which of elements are selected randomly at kth sector of jth region on target data, and V is value set corresponding to $O_{j,k}$ (See Figure 2). If only one value of data storage does not correspond to that of target data, the algorithm decides that they are different. Therefore, through searching by region unit in reverse order, the number of comparison operations should be cut down. The fact that file slack could be detected is another advantage of this approach. If the fragments of target data remain at the back end of the cluster, this method can identify them.

This suggested method makes FNR zero as it compares all sectors of data storage and target file. When this method compares two sectors, the collision resistance of the proposed method is theoretically $1/2^{128}$ which is the same as MD5. In the random offset sampling, a phase of hash calculation is not required, so this method has merit in the processing time relatively. The effectiveness of the proposed algorithm is proved through an experiment in Section V.

Nevertheless, this method has a risk factor that the collision resistance is based on the assumption; the probability distribution of byte values for each offset is uniform distribution. If the distribution is not uniform, the collision resistance cannot guarantee $1/2^{128}$. This fact can be prompt considerable false positive. In the next section, we figure out the real distribution through analyzing storage mediums actually in use and verify whether the assumption is reasonable or not.

IV. PROPOSED METHOD 2 : STATISTICAL OFFSET SAMPLING

To verify the real probability distribution of byte values for each offset, we investigated 50 storage mediums. Table 1 shows the medium's specifications such as capacity, file

TABLE 1. Experimental storage mediums.

Media	Device Info.	Description	Media	Device Info.	Description
SSD1 (a)	Samsung	1TB, NTFS	USB5	Sandisk	16GB, NTFS
SSD2 (b)	Micron	512GB, NTFS, OS installed	USB6	Sandisk	8GB, NTFS
SSD3	Samsung	256GB, NTFS, OS installed	USB7	Sandisk	64GB, Ext4
SSD4	Samsung	128GB, NTFS	USB8	Sandisk	64GB, exFAT
SSD5	Sandisk	256GB, exFAT	USB9	Sandisk	64GB, NTFS
SSD6	Sandisk	256GB, exFAT	USB10	Sandisk	128GB, NTFS
SSD7	Western Digital	256GB, Ext4, OS installed	USB11	Samsung	64GB, exFAT
SSD8	Mystor	128GB, NTFS, OS installed	USB12	Samsung	64GB, NTFS
SSD9	crucial	250GB, NTFS, OS installed	USB13	Samsung	64GB, NTFS
HDD1 (c)	Toshiba	512GB, NTFS	USB14	HP	16GB, FAT32
HDD2	Western Digital	2TB, NTFS, OS installed	USB15	Transcend	32GB, exFAT
HDD3	Hitachi	1TB, NTFS, OS installed	USB16	memorette	4GB, FAT32
HDD4	Hitachi	512GB, NTFS	USB17	memorette	16GB, NTFS
HDD5	Seagate	512GB, exFAT	SmartPhone1 (e)	Galaxy S6	25.8GB, Ext4
HDD6	Seagate	512GB, exFAT	SmartPhone2	Galaxy S7	29.7GB, Ext4
HDD7	Seagate	1TB, Ext4	SmartPhone3	Vega Iron2	29.1GB, Ext4
HDD8	Western Digital	512GB, NTFS	SmartPhone4	Galaxy Note 2	53.6GB, Ext4
HDD9	Western Digital	1TB, NTFS	SmartPhone5	Galaxy S3	29.1GB, Ext4
HDD10	Toshiba	512GB, NTFS	SD Card1 (f)	Black box	8GB, NTFS
HDD11	Toshiba	1TB, NTFS	SD Card2	Galaxy S3	16GB, Ext4
HDD12	Seagate	1TB, NTFS	SD Card3	Galaxy S7	16GB, Ext3
USB1 (d)	Transcend	1GB, NTFS	SD Card4	Galaxy S6	32GB, FAT32
USB2	ADATA	1.9GB, FAT32	SD Card5	Black box	32GB, FAT32
USB3	HP	8GB, FAT32	SD Card6	Black box	32GB, FAT32
USB4	Sandisk	128GB, exFAT	SD Card7	Black box	32GB, FAT32

system, medium type. The mediums have been connected to PC and used by people. We examined the frequency of the bye values for specific offset and documented it as a ratio from 0 to 1. It can be described as $P_o(v)$ that o and v respectively represent specific offset in a sector and byte value. The range of o is from 0 to 511 since the size of the sector is 512 bytes and the range of v is from 0×0 to $0 \times$ FF since it represents 1 byte. To identify a shape of the distribution on storage devices, we drew a graph chart for 0∼511 offsets. Figure 4 shows an example of a storage medium (a) which is described in Table 1.

If the distribution follows a uniform distribution, all probability would be $1/256$ and the appearance of a graph would be an even shape. However, as seen in Figure 4, the shape of distribution extremely different from flat shape. It is observed that the probability of 0×00 value is extremely higher than other values for all offsets and the $0 \times$ FF's probability tends to be higher than others. Similar to the graph for medium (a), we drew graphs for the experimental storage mediums. Figure 4 shows the graphs of 6 mediums marked in Table 1 as an example. Including the 6 mediums, all graphs are analogous in that the probability of 0×00 value is greater than other values. To identify the graph shape of offsets except 0×00 value, we regarded the frequency of 0×00 as an outlier and drew the graphs like Figure 5.

Though the 0×00 values are removed on the charts, it is visible to the naked eye that there are wide variations; it means that the distribution of offset-value is completely different from the uniform distribution. The graphs of (a), (c), and (d) shows different shape though they are NTFS without Windows OS. We presume that the variety comes from the user's intent. For example, the user may use the 1TB HDD for storage of video files as they generally require large capacity, on the other hand, the 512GB HDD may be used for recording photos, presentation files, etc. Note, the graphs for the rest of the mediums not included in the figure also show a similar appearance. Based on this knowledge, we improved the random offset sampling by not arbitrarily but statistically selecting the 16 offsets through calculating the probability of value's ratio for each offset. If the 16 offset-values with less frequency can be chosen, FPR would be significantly dropped. To advance our method, we verified the real distribution of byte value with the experimental data.

A. PROBABILITY DISTRIBUTION OF BYTE VALUE'S RATIO

The probability distribution of the byte value's ratio doesn't follow the known probability distribution shape including the uniform distribution or normal distribution as seen in Figure. 4 and Figure 5. Instead, we figured out that the distribution on the same medium has similar in appearance and medium's distributions look a bit similar pattern, for example, the frequency of 0×00 and $0 \times$ FF values tends to higher than others. To test that the characteristics are statistically robust, we tested the hypothesises by the experimental data from storage mediums.

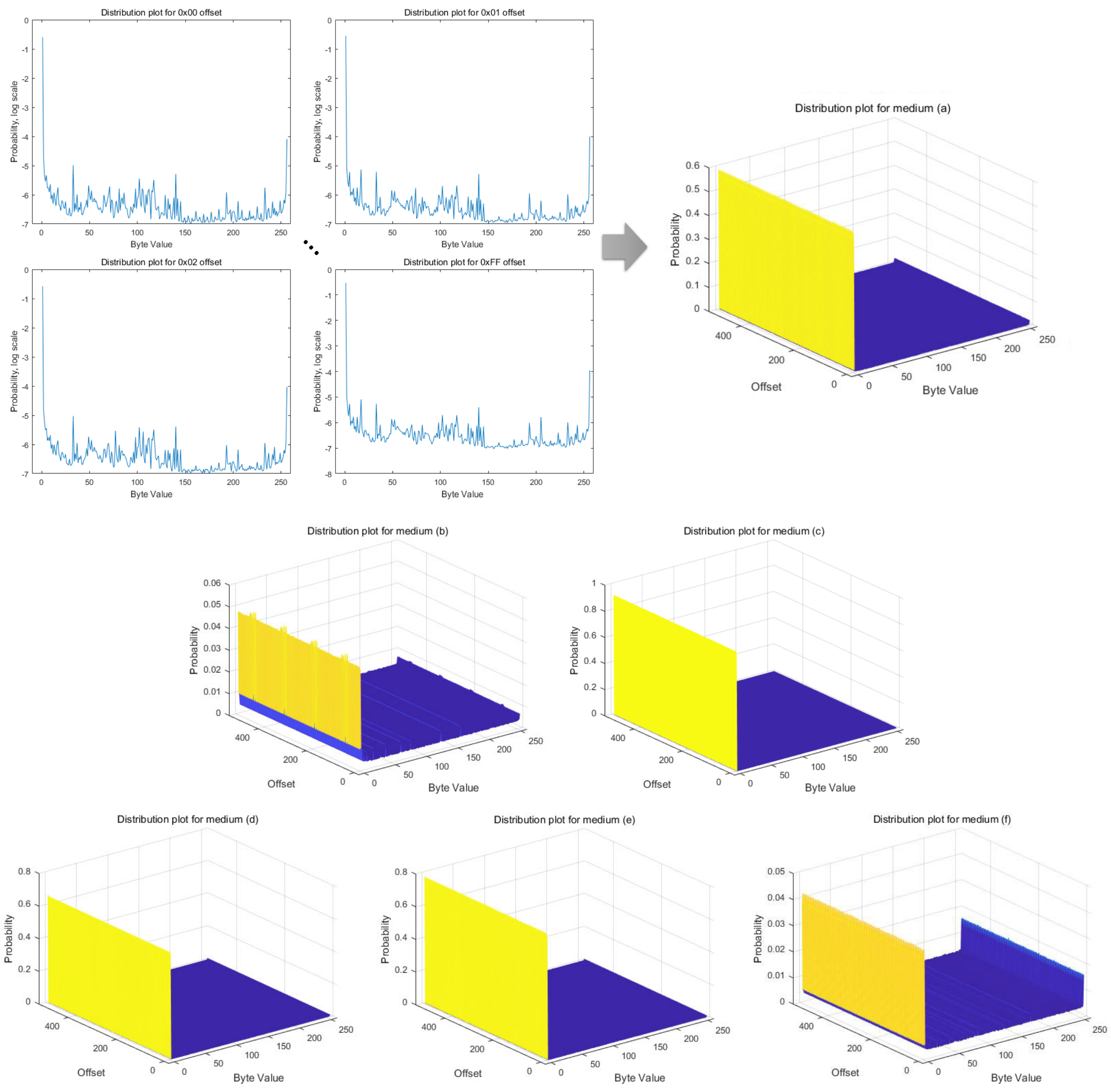


FIGURE 4. Example of the graph of values' distribution for all offsets on a storage medium (a) and the offset-value distribution of 6 storage mediums.

Firstly, the Kolmogorov-Smirnov test is used to test whether the distributions of value's ratio for each offset come from the same distribution [37]. We chose the (a)∼(f) mediums marked in Table 1 as sample and $130,816 (= {}^{512}C_2)$ tests were performed per one medium as there are 512 distributions. The null hypothesis was that the distributions on arbitrary two offsets come from the same distribution and the significance level was 5%. As a result of the test, the null hypothesis was rejected in 130,816 tests out of a total $784,896 (= 130,816 \times 6)$ tests; 41,210 tests on medium (a), 34,515 tests on medium (b), 7,845 tests on medium (c), 10,014 tests on medium (d), 21,471 tests on medium (e), and 15,761 tests on medium (f) were rejected. Through this result, we verified that the assumption that the distributions are from the same distribution is not valid.

In sequence, we tested the hypothesis that the byte values for specific offsets are from the same distribution regardless of mediums. Similar to the previous test, the 6 mediums were used as a sample and a total of $7,680 (= {}^{6}C_2 \times 512)$ tests were examined. The Kolmogorov-Smirnov test was also

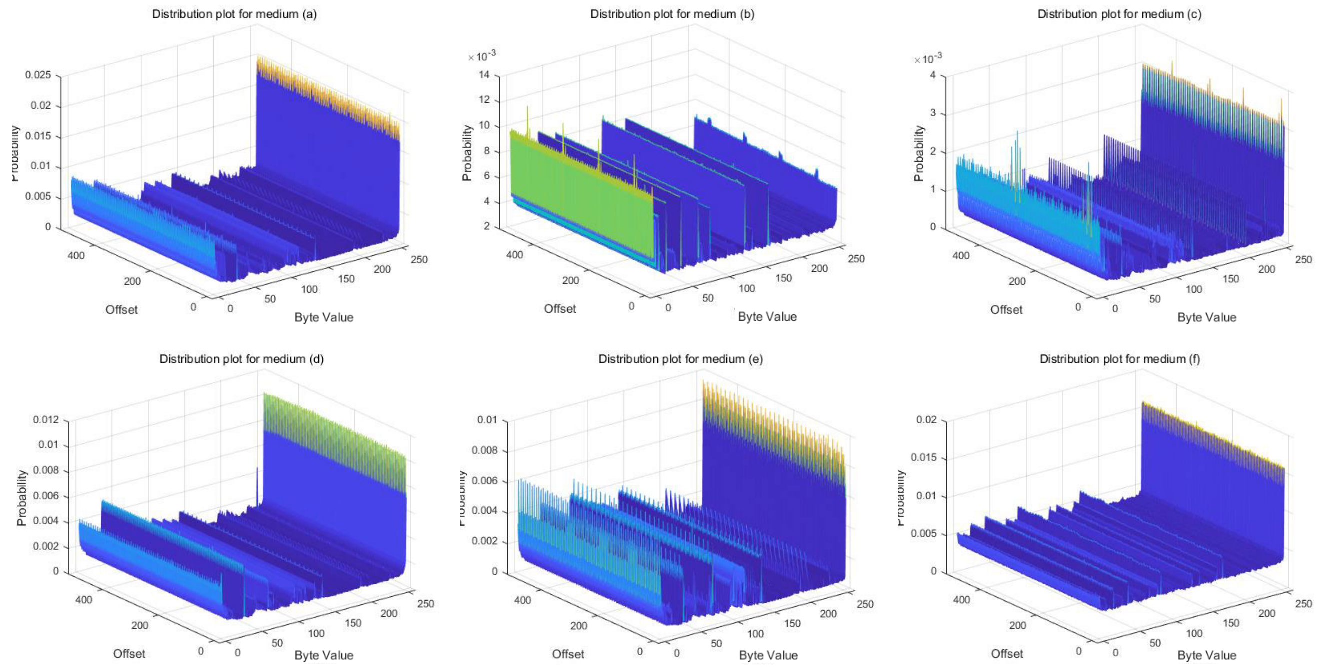


FIGURE 5. The Offset-value distribution of 6 storage mediums without the 0 × 00 values.

performed at the 5% significance level. As a result, since 7,670 tests were rejected so the assumption was also rejected.

Finally, we estimated a population ratio of specific byte value on specific offset. To check whether the distributions satisfy the central limit theorem, we performed the normality test. We chose the 30 devices from the experimental storage mediums randomly and then we calculated the mean of the ratio a hundred times. For example from chosen the 30 devices, the ratio of the 0×00 value on the 0×00 offset is selected and then the mean of the ratios is calculated. Figure 6 shows the histogram of the means of the $0 \times 00 \sim 0 \times 09$ values on the 0×00 offset. As seen in the histogram, the sample means for each value follows approximately the normal distribution so it can be verified that the values for specific offsets satisfy the central limit theorem. Based on the theorem, we calculated the mean of the sample means as seen in Table 2. The values in the table show population ratio of offset-value because the mean of the sample means is approximately close to the population mean. The ratio of 0×00 was extremely higher on the medium as the unallocated area remains 0×00 due to the default value of the data storage saved as 0×00. Another characteristic found was that $0 \times FF$ frequency was detected higher than any other figure except 0×00. We inferred that the unallocated area in a file such as the unused area is mostly filled by $0 \times FF$ (See Figure 7). The fact that $0 \times FF$ is used as end of file marker or end of record marker in the NTFS MFT can be another possible reason for the many occurrences.

B. OFFSET SAMPLING

Based on the calculated distribution from the previous section, the 16 offsets with the smallest expected probability can be selected as shown in the following equation.

$$\{(o^*, v)\} = arg \min_{(o,v)} \prod_{i=1}^{16} P_o(v) \tag{2}$$

The o means the offset, and v means the byte value which is stored on the o. P_o indicates the probability of the v at the o. Likewise the random offset sampling, the target data is inputted and then the 16 offset-values are chosen for comparison. The method to search on the target data is equal to the method suggested in Section III. So, this method searches the target data by the region unit and the sectors, inside the region, are examined in reverse order. By comparing the selected 16 offset-values, the identical sectors can be detected. Since the suggested method employs the complete enumeration, the FNR gets 0. The FPR is expected to be lower compared to the proposed method 1 as this method uses statistical sampling.

V. EXPERIMENT

We tested empirically the four algorithms: traditional hash comparison method by sector unit, random sampling method suggested by [11], random offset sampling, and statistical offset sampling. The equation of FNR, collision resistance, and time for the four algorithms are represented in Table 3. t represents the data size of the target. d represents the size of the data storage. T_h represents the time for calculating the MD5 hash value. T_c represents the time for comparing the two 16 bytes strings. T_p represents the time for calculating the equation 2. $P(t)$ represents the percentage of samples needed for examining storage media based on the size of target data suggested in [11]. Note, the collision resistance of the random offset sampling is the theoretical value based

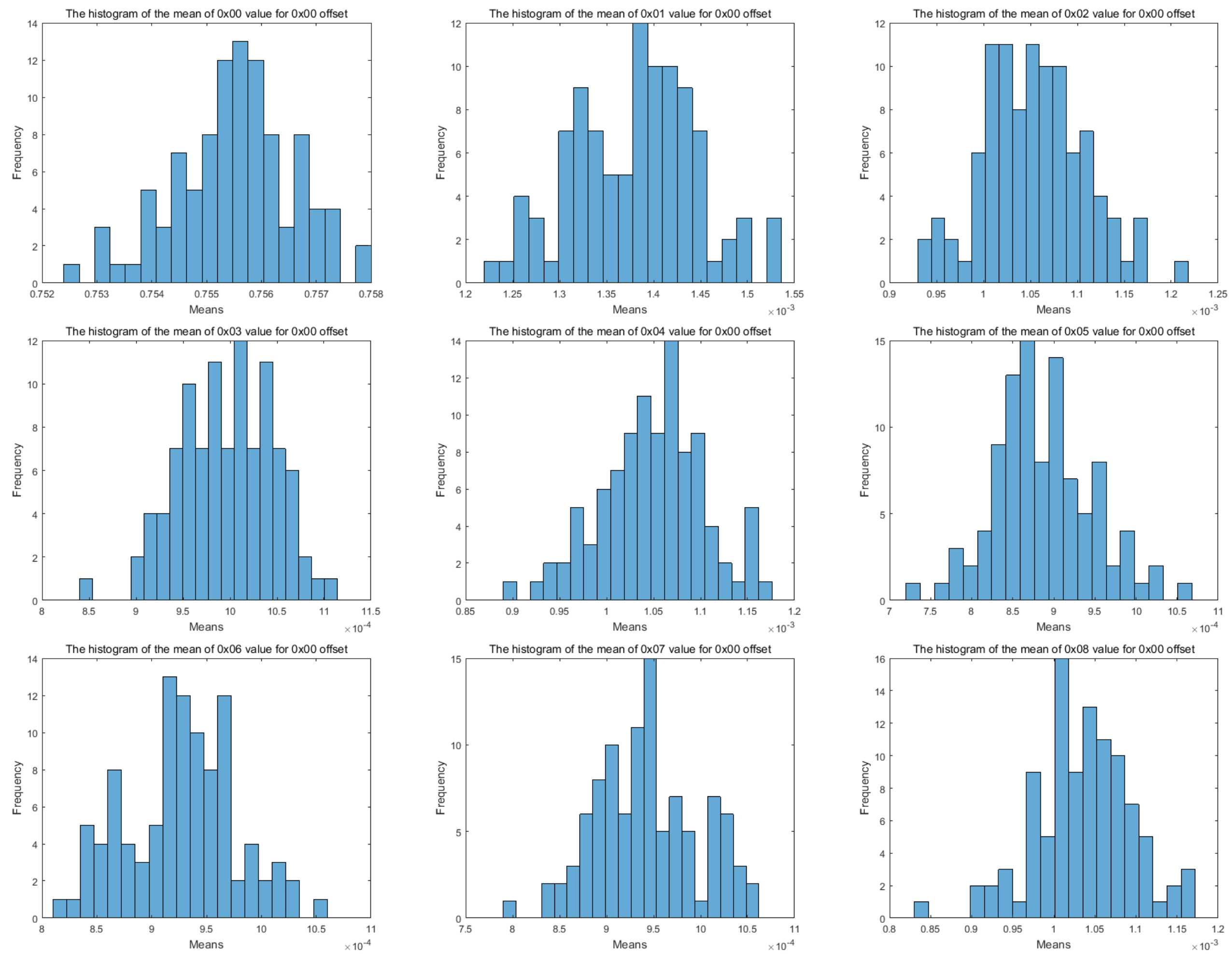
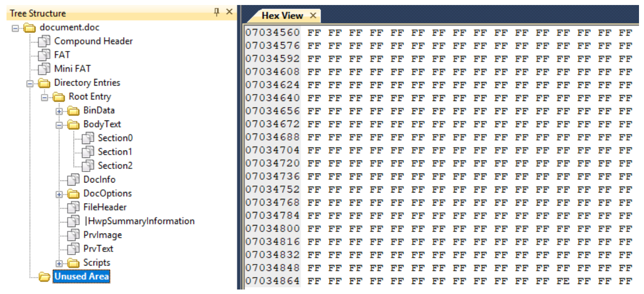


FIGURE 6. **The histogram of the mean of values on the 0 × 00 offset (n=30).**



```
07034560 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
07034576 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
07034592 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
07034608 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
07034624 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
07034640 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
07034656 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
07034672 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
07034688 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
07034704 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
07034720 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
07034736 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
07034752 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
07034768 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
07034784 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
07034800 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
07034816 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
07034832 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
07034848 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
07034864 FF FF FF FF FF FF FF FF FF FF FF FF FE FF FF FF
```

FIGURE 7. **Example of unused area inside a document file.**

on the assumption that the byte values stored in the medium follow the uniform distribution. Collision resistance of statistical offset sampling shows the minimum and the maximum based on equation 2. For example, the collision resistance can be $1/10^2$ when all byte values of a region are 0 entirely.

A. EXPERIMENT METHOD

We tested 256GB (536,870,912 sectors) SSD and 1TB (2,147,483,648 sectors) HDD actually in use. Before testing algorithms, we stored five files as target data: sample.docx with the size of 4.35MB (8,908 sectors), sample.jpg with the size of 8.14MB (16,671 sectors), sample.exe with the size of 80.4MB (164,660 sectors), sample.msi with the size of 149MB (305,152 sectors), and sample.mp4 with the size of 487MB (997,376 sectors). On the SSD, each file was recorded consecutively, in other words, they were not fragmented. The data of the docx, exe, msi on the HDD were also not fragmented, however, the jpg and mp4 were fragmented in three parts and twenty-three parts.

When it comes to random sampling, the sample size was determined by previous research. By the [11], at least the one sector of the target file can be found by 0.99999 percent chance through sampling the 13.5, 6.7, 0.84, 0.42, 0.1 percent of the medium when the size of the target data is 4MB, 8MB, 64MB, 128MB, and 512MB respectively. The region size was set as 4KB for both random offset sampling and statistical offset sampling. We used experimental systems that the processor was Intel i7-4790 and the memory capacity was 32GB. In this experiment, we used only one single thread, so the

TABLE 2. Table of the probability distribution of offset-value.

Offset	0x00	0x01	0x02	0x03	0x04	...	0xFB	0xFC	0xFD	0xFE	0xFF
0	0.755632	0.00138	0.001056	0.000999	0.00105	...	0.000957	0.001207	0.001043	0.00102	0.002671
1	0.756591	0.00134	0.00105	0.001076	0.001137	...	0.00102	0.001325	0.001019	0.001	0.002602
2	0.755728	0.001643	0.001086	0.001036	0.001075	...	0.000917	0.00131	0.001011	0.000972	0.002649
3	0.757120	0.001303	0.001044	0.001053	0.001126	...	0.000940	0.001284	0.001008	0.001043	0.002738
4	0.755966	0.001337	0.001106	0.001032	0.001035	...	0.000918	0.001255	0.000925	0.001059	0.002632
...	...	...	...	...	...	...	...	...	...	...	...
507	0.758724	0.001301	0.001046	0.001064	0.001117	...	0.000921	0.001251	0.001007	0.001006	0.002678
508	0.757674	0.001382	0.001066	0.001022	0.001027	...	0.000877	0.001293	0.000918	0.001047	0.002592
509	0.758586	0.00132	0.001068	0.001062	0.001107	...	0.000854	0.001254	0.000943	0.001009	0.002539
510	0.75776	0.001371	0.001134	0.001168	0.001113	...	0.000905	0.001165	0.001045	0.000969	0.002501
511	0.758985	0.001314	0.001041	0.001078	0.0011	...	0.000923	0.001309	0.000989	0.001013	0.002705

TABLE 3. Algorithm comparison.

	Hash comparison of sectors	Random sampling [11]
Collision Resistance	$\frac{1}{2^{128}}$	$\frac{1}{2^{128}}$
False Negative Rate	0	Depend on sample size
Processing Time	$T_h(t) + T_h(d) + T_c(t \times d)$	$T_h(t) + T_h(d \times P(t)) + T_c(t \times d \times P(t))$
	Random offset sampling	**Statistical offset sampling**
Collision Resistance	$\frac{1}{2^{128}}$	$\frac{1}{4.03 \times 10^{51}} \sim \frac{1}{10^2}$
False Negative Rate	0	0
Processing Time	$T_c(t \times d)$	$T_p(t) + T_c(t \times d)$

processing speed can be faster than this experiment if using multi-thread [18].

B. RESULT - TIME

Table 4 shows the processing time of algorithms. Statistical offset sampling took more time in the initialization phase than other methods as it should calculate probability before selecting the 16 offsets. However, as the search goes on, the search time was rapidly reduced compared to other methods. Compared to random offset sampling, the time gets relatively reduced as the size of data storage increases. Although two methods are equal in the data search order, statistical offset sampling can detect the unmatched sectors faster than random offset sampling because the former one compares the selected offsets of statistically low frequency. In other words, the number of comparison operation has declined and this fact affects the processing time. Especially, when the target data was relatively small like the DOCX, the statistical offset sampling method took less time, compared to using random sampling as seen in Figure 8. This is because reading disk spent less time than calculating the hash for 13.5 percent of data storage and comparing the hash list.

Another remarkable thing was the processing time of random sampling. Regardless of the size of the target data, it showed a similar time performance. Other methods require more comparison operation as target data increase in size. Especially hash comparison method was impractical when the size of the target data was more than 80MB. This is because the three methods except random sampling adopt complete enumeration so the processing time increases linearly.

C. RESULT - ACCURACY

The accuracy of the algorithm is as much important as time in digital forensics. Specifically, forensic examiners are sensitive to false-negative as they can miss critical evidence. Table 5 shows the experimental results including accuracy for each method and target. Note, as seen in Table 4, the algorithms except the random sampling required a considerable time, so we compared the results of docx, jpg, and exe file. True-positive (TP), true-negative (TN), false-positive (FP), and false-negative (FN) in the table means the number of the sector found in the medium. The total number of sectors of media and target file can be seen in Section V-A.

The notable factor is that TP of random sampling is lower and FN of the same method is dramatically higher than the other methods; this is bound to be the algorithm characteristics. Specifically, any fragment of docx file was not detected as this results from the characteristics of how the file is stored in the medium [36]. Generally, data are saved in the consecutive sector when the file is stored in the data storage. In this experiment, the size of the data storage is 256GB while the size of docx file is 4MB that is 1.5×10^{-5} percentage of the data storage. If the algorithm passes by the location where the data is saved, it will no longer be found since the target file is saved in consecutive order. In the case of jpg file, it detected sectors in two fragments out of three. These results imply that the random sampling is suitable to not find all fragments of the target but identify the existence of the target file rapidly. The other methods detect the entire sectors regardless of the target type as they conduct complete enumeration. All algorithms showed FP more than 0. Our analysis found some parts of the target file, like Figure 7, were the same to the unallocated area of other files by chance.

TABLE 4. The result of the experiment - Time (second). Note, the * mark means the location of the target file's last sector. When the processing time is longer than 14 days, the estimated time is recorded.

Data Storage	Target	Volume	Hash comparison	Random sampling	Random offset sampling	Statistical offset sampling
SSD 256GB	DOCX	0	0.09	0.09	0.17	2.15
		10GB	14028.89	1992.10	2120.64	1398.62
		76GB*	106598.10	14284.15	16232.83	10454.28
		250GB	341384.58	47452.46	49621.01	31027.13
	JPG	0	0.11	0.11	0.35	3.98
		10GB	22937.71	1536.83	3852.10	2639.18
		100GB	226555.76	15858.90	37519.41	25441.71
		249GB*	563668.47	38385.82	87795.43	56989.43
	EXE	0	1.05	1.05	3.45	38.36
		10GB	230229.56	1933.93	39726.27	26249.59
		100GB	≃ 26 days	19333.11	384153.01	247008.61
		177GB*	≃ 47 days	34408.66	668426.24	405094.12
		250GB	≃ 66 days	47946.11	930867.92	582343.42
	MSI	0	1.68	1.68	6.15	72.53
		10GB	422913.68	1776.24	72931.79	48042.87
		100GB	≃ 48 days	18184.74	722024.74	463133.24
		217GB*	≃ 106 dyas	39462.08	≃ 17 days	917003.81
	MP4	0	5.49	6.36	19.71	244.05
		5GB	666219.89	666.22	116456.97	74212.90
		77GB*	≃ 118 days	10259.79	≃ 20 days	1098350.98
HDD 1TB	DOCX	0	0.09	0.09	0.20	2.16
		10GB	14233.69	2035.42	2263.93	1517.12
		100GB	145382.91	19626.69	22186.54	13805.84
		640GB*	931904.45	125807.10	199678.84	121608.39
	JPG	0	0.11	0.11	0.35	3.99
		10GB	26419.31	1796.51	4112.33	2617.72
		100GB	263868.14	18734.64	40136.34	23559.44
		229GB*	656424.78	44965.10	94320.39	54186.70
	EXE	0	1.08	1.08	3.62	38.37
		10GB	230503.48	1936.23	40707.15	26577.90
		100GB	≃ 26 days	19212.37	394859.32	250629.58
		500GB*	≃ 132 days	96711.23	≃ 22 days	1102770.16
	MSI	0	1.85	1.85	6.56	73.00
		10GB	455681.85	1918.42	81966.70	48869.71
		100GB	≃ 52 days	19568.85	818027.70	472081.40
		217GB*	≃ 114 days	42539.03	≃ 19 days	939441.99
	MP4	0	6.05	6.05	20.20	249.40
		10GB	≃ 15 days	1346.16	227122.83	140534.83
		100GB	≃ 155 days	13623.10	≃ 25 days	9154754.06
		756GB*	≃ 1177 days	101769.43	≃ 187 days	≃ 105 days

TABLE 5. The result of the experiment - Accuracy. Note, the TP means how many sectors of the target file are detected.

Method	Target	TP	TN	FP	FN	precision	recall	accuracy	specificity	F1
Hash comparison	DOCX	8,908	536,861,970	34	0	0.9962	1.0000	1.0000	1.0000	0.9981
	JPG	16,671	536,851,824	2,417	0	0.8734	1.0000	1.0000	1.0000	0.9324
	EXE					N/A				
Random sampling	DOCX	0	536,682,000	4	8,908	0.0000	0.0000	1.0000	1.0000	-
	JPG	1,157	536,854,107	134	15,514	0.8962	0.0694	1.0000	1.0000	0.1288
	EXE	987	536,706,252	0	163,673	1.0000	0.0060	0.9997	1.0000	0.0119
Random offset sampling	DOCX	8,908	536,861,955	49	0	0.9945	1.0000	1.0000	1.0000	0.9973
	JPG	16,671	536,848,094	6,147	0	0.7306	1.0000	1.0000	1.0000	0.8443
	EXE	164,660	536,684,678	21,574	0	0.8842	1.0000	1.0000	1.0000	0.9385
statiscal offset sampling	DOCX	8,908	536,861,970	40	0	0.9955	1.0000	1.0000	1.0000	0.9978
	JPG	16,671	536,851,824	2,971	0	0.8487	1.0000	1.0000	1.0000	0.9182
	EXE	164,660	536,851,824	1,471	0	0.9911	1.0000	1.0000	1.0000	0.9956

On the other hand, random offset sampling and statistical offset sampling had a difference in FP but not in TP. The FP occurs when 16 selected offsets from the target file are the same as the other file or the unallocated area of the file system by chance. Since the FP of random offset sampling was higher than the one of statistical offset sampling, the F1 score of random offset sampling was observed lower than the one of statistical offset sampling.

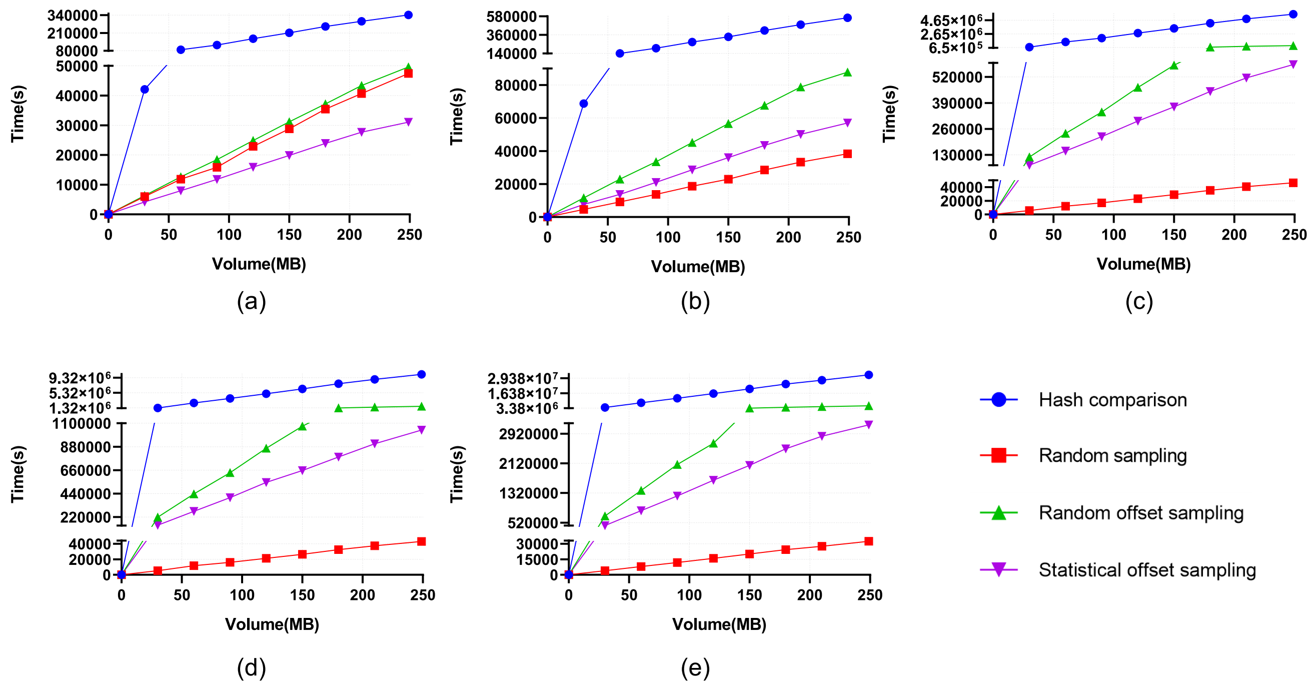


FIGURE 8. Comparison of the processing time of 4 methods on SSD. (a) sample.docx (b) sample.jpg (c) sample.exe (d) sample.msi (e) sample.mp4. The values are described in Table 4.

D. DISCUSSION

Through the experiment, we verified the performance of the algorithms and learned how to use the proposed method and previous researches correctly when finding target data on data storage. We wish to share the following lessons:

- The 16 bytes extracted from the sector can be used as an alternative to cryptographic hash when a forensic examiner wants to find target data on data storage.
- If the bytes are selected statistically, it is more powerful to find a fragment of the target file than randomly selection.
- The proposed method shows good performance to identify all traces of the target file. However, if the goal is to check for only the existence of the fragment of the target file and the size of the target file is over 10MB, random sampling will show the best performance.
- Applying our method to existing random sampling methods will give better performance as hash operations are not required.

VI. CONCLUSION

In this paper, we have proposed a new approach which mixes the advantages of complete enumeration and random sampling. Our proposed method searches all areas of data storage entirely and compares with target data, as a result, false-negative gets zero. To overcome the limitation of complete enumeration that imposes an extensive delay in the investigation process, we used 16 bytes in a sector between a storage device and a target file instead of hash comparison. The byte values are chosen statistically based on the probability distribution estimated by the central limit theorem, so we can determine whether two sectors are identical or not using the values with high accuracy. This method searches data by region unit instead of the sector; the processing time is significantly reduced. We proved that the proposed method can be used as an alternative to previous methods through the empirical experiment. We also verified that the true negative is very close to hash comparison and the processing speed can be faster than random sampling in particular cases.

A limitation of our method is that the processing time increases linearly as the size of the target data increases. This method can not deal with semantically identical files and encrypted data. Another drawback is that an interpretation of accuracy can be misled if the distribution deduced from an imbalanced data-set.

Despite this limitation, the result of this study can provide a significant contribution to digital forensics as the proposed method can support previous researches. This paper can inspire forensic researchers and examiners as our concept is applicable to other sub-fields of digital forensics such as data carving, forensic triage, abnormality detection, slack detection, etc. The future work is to clustering devices and profiling an individual based on the proposed algorithm. We will also apply our methodology to identifying specific file formats such as video, document, graphics, etc.

REFERENCES

[1] M. R. Grossman and G. V. Cormack, "Technology-assisted review in e-discovery can be more effective and more efficient than exhaustive manual review," *Richmond J. Law Technol.*, vol. 17, no. 3, p. 1, 2010.

[2] D. W. Oard and W. Webber, "Information retrieval for e-discovery," *Found. Trends Inf. Retr.*, vol. 7, nos. 2–3, pp. 99–237, 2013.
[3] F. Breitinger, B. Guttman, M. McCarrin, V. Roussev, and D. White, "Approximate matching: Definition and terminology," *NIST Special Publication*, vol. 800, p. 168, May 2014.
[4] L. Caviglione, S. Wendzel, and W. Mazurczyk, "The future of digital forensics: Challenges and the road ahead," *IEEE Security Privacy*, vol. 15, no. 6, pp. 12–17, Nov./Dec. 2017.
[5] J. Lee and S. Un, "Digital forensics as a service: A case study of forensic indexed search," in *Proc. Int. Conf. ICT Converg. (ICTC)*, Oct. 2012, pp. 499–503.
[6] T. M. Paixão, M. C. Boeres, C. O. A. Freitas, and T. Oliveira-Santos, "Exploring character shapes for unsupervised reconstruction of strip-shredded text documents," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 7, pp. 1744–1754, Jul. 2019.
[7] Z. Boulkenafet, J. Komulainen, and A. Hadid, "Face spoofing detection using colour texture analysis," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 8, pp. 1818–1830, Aug. 2016.
[8] S. L. Garfinkel and M. McCarrin, "Hash-based carving: Searching media for complete files and file fragments with sector hashing and hashdb," *Digit. Invest.*, vol. 14, pp. S95–S105, Aug. 2015.
[9] D. Quick and K.-K. R. Choo, "Impacts of increasing volume of digital forensic data: A survey and future research challenges," *Digit. Invest.*, vol. 11, no. 4, pp. 273–294, Dec. 2014.
[10] S. L. Garfinkel, "Digital forensics research: The next 10 years," *Digit. Invest.*, vol. 7, pp. S64–S73, Aug. 2010.
[11] N. K. Bharadwaj and U. Singh, "Efficiently searching target data traces in storage devices with region based random sector sampling approach," *Digit. Invest.*, vol. 24, pp. 128–141, 2018.
[12] J. K. Taguchi, "Optimal sector sampling for drive triage," Naval Postgraduate School, Monterey, CA, USA, Tech. Rep., 2013.
[13] N. Canceill, "Random sampling applied to rapid disk analysis," Tech. Rep., 2013.
[14] J. Kornblum, "Identifying almost identical files using context triggered piecewise hashing," *Digit. Invest.*, vol. 3, pp. 91–97, Sep. 2006.
[15] V. Roussev, "Data fingerprinting with similarity digests," in *Proc. IFIP Int. Conf. Digit. Forensics*. Springer, 2010, pp. 207–226.
[16] F. Breitinger and H. Baier, "Similarity preserving hashing: Eligible properties and a new algorithm MRSH-v2," in *Proc. Int. Conf. Digit. Forensics Cyber Crime*. Springer, 2012, pp. 167–182.
[17] P. Penrose, W. J. Buchanan, and R. Macfarlane, "Fast contraband detection in large capacity disk drives," *Digit. Invest.*, vol. 12, pp. S22–S29, Mar. 2015.
[18] S. McKeown, G. Russell, and P. Leimich, "Sub-file hashing strategies for fast contraband detection," in *Proc. Int. Conf. Cyber Secur. Protection Digit. Services*, Jun. 2018, pp. 1–7.
[19] L. F. da Cruz Nassif and E. R. Hruschka, "Document clustering for forensic analysis: An approach for improving computer inspection," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 1, pp. 46–54, Jan. 2013.
[20] N. L. Beebe, L. A. Maddox, L. Liu, and M. Sun, "Sceadan: Using concatenated N-gram vectors for improved file and data type classification," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 9, pp. 1519–1530, Sep. 2013.
[21] F. Iqbal, H. Binsalleeh, B. C. M. Fung, and M. Debbabi, "Mining writeprints from anonymous e-mails for forensic investigation," *Digit. Invest.*, vol. 7, nos. 1–2, pp. 56–64, 2010.
[22] S. Decherchi, S. Tacconi, J. Redi, A. Leoncini, F. Sangiacomo, and R. Zunino, "Text clustering for digital forensics analysis," in *Computational Intelligence in Security for Information Systems*. Springer, 2009, pp. 29–36.
[23] G. G. Dagher and B. C. M. Fung, "Subject-based semantic document clustering for digital forensic investigations," *Data Knowl. Eng.*, vol. 86, pp. 224–241, Jul. 2013.
[24] N. L. Beebe and L. Liu, "Clustering digital forensic string search output," *Digit. Invest.*, vol. 11, no. 4, pp. 314–322, Dec. 2014.
[25] V. SaiKrishna, A. Rasool, and N. Khare, "String matching and its applications in diversified fields," *Int. J. Comput. Sci. Issues*, vol. 9, no. 1, p. 219, 2012.
[26] R. Caldelli, I. Amerini, F. Picchioni, and M. Innocenti, "Fast image clustering of unknown source images," in *Proc. IEEE Int. Workshop Inf. Forensics Secur.*, Dec. 2010, pp. 1–5.
[27] I. Amerini, R. Caldelli, P. Crescenzi, A. D. Mastio, and A. Marino, "Blind image clustering based on the Normalized Cuts criterion for camera identification," *Signal Process. Image Commun.*, vol. 29, no. 8, pp. 831–843, 2014.
[28] L. G. Villalba, A. S. Orozco, and J. R. Corripio, "Smartphone image clustering," *Expert Syst. Appl.*, vol. 42, pp. 1927–1940, Sep. 2015.
[29] X. Lin and C.-T. Li, "Large-scale image clustering based on camera fingerprints," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 4, pp. 793–808, Apr. 2017.
[30] M. K. Rogers, J. Goldman, R. Mislan, T. Wedge, and S. Debrota, "Computer forensics field triage process model," *J. Digit. Forensics, Secur. Law*, vol. 1, no. 2, p. 2, 2006.
[31] V. Roussev, C. Quates, and R. Martell, "Real-time digital forensics and triage," *Digit. Invest.*, vol. 10, no. 2, pp. 158–167, Sep. 2013.
[32] S. L. Garfinkel, "Digital media triage with bulk data analysis and bulk_extractor," *Comput. Secur.*, vol. 32, pp. 56–72, Feb. 2013.
[33] G. Horsman, C. Laing, and P. Vickers, "A case-based reasoning method for locating evidence during digital forensic device triage," *Decis. Support Syst.*, vol. 61, pp. 69–78, May 2014.
[34] D. Quick and K.-K. R. Choo, "Big forensic data reduction: Digital forensic images and electronic evidence," *Cluster Comput.*, vol. 19, no. 2, pp. 723–740, 2016.
[35] S. Garfinkel, A. Nelson, D. White, and V. Roussev, "Using purpose-built functions and block hashes to enable small block and sub-file forensics," *Digit. Invest.*, vol. 7, pp. S13–S23, Aug. 2010.
[36] M. Hirano, H. Takase, and K. Yoshida, "Evaluation of a sector-hash based rapid file detection method for monitoring infrastructure-as-a-service cloud platforms," in *Proc. 10th Int. Conf. Availability, Rel. Secur.*, Aug. 2015, pp. 584–591.
[37] L. Al-Labadi and M. Zarepour, "Two-sample Kolmogorov-Smirnov test using a Bayesian nonparametric approach," *Math. Methods Statist.*, vol. 26, no. 3, pp. 212–225, 2017.

DOOWON JEONG received the B.S. degree from the Division of Industrial Management Engineering, Korea University, in 2011, and the Ph.D. degree from the Graduate School of Information Security, Korea University, in 2019. He is currently a Research Professor with the Division of Information Security, Korea University. His research interests include digital forensics, information security, and digital profiling.

SANGJIN LEE received the Ph.D. degree from the Department of Mathematics, Korea University, in 1994. From 1989 to 1999, he was a Senior Researcher with the Electronics and Telecommunications Research Institute, South Korea. He has been running the Digital Forensic Research Center, Korea University, since 2008. He is currently the President of the Division of Information Security, Korea University. He has authored or coauthored over 130 articles in various archival journals and conference proceedings and over 200 articles in domestic journals.

His research interests include digital forensics, data processing, forensic framework, incident response, and so on.