Advanced Science and Technology Letters
Vol.28 (CIA 2013), pp.82-87
http://dx.doi.org/10.14257/astl.2013.28.16

LwVLC : Light-weight Variable-length Chunking Scheme for File Similarity Search in Digital Forensics

Min Ja Kim[1], Chuck Yoo[1], Wan Yeon Lee[3.] ,Young Woong Ko[2]

[1]Dept. of Computer Science and Engineering, Korea University, Seoul, Korea
{mjfeel,hxy}@korea.ac.kr
[2]Dept. of Computer Engineering, Hallym University Chuncheon, Korea
yuko@hallym.ac.kr
[3]Dept. of Computer Science, Dongduk Womens University, Seoul, Korea
wanlee@dongduk.ac.kr

Abstract. Forensic examination of computer file system is an important component of cyber-security. As the capacity of storage devices are continuously growing, it leads to difficulties and time-consuming task to examine lots of files. Therefore, if we find an effective way to search file similarity, large amounts of files can be analyzed within the time constraints of an investigation and the overall performance of file forensics can be improved. In this paper, we examine a novel scheme for approximate file similarity search based on VLC hash scheme which is light-weight variable-length chunking scheme for file similarity search in digital forensics (LwVLC). The key idea is to minimize metadata size of file similarity information and reduce the execution time of similarity search with high accuracy.

Keywords: Digital forensics, File similarity, Fixed-length chunking, Variable-length chunking, Metadata, hash key

1 Introduction

Generally file system analysis is the most common and easily performed for forensic examination of computers. An important source of malicious information is data found in various files on a typically large storage devices. Analyzing large amounts of data effectively within the time constraints of an investigation poses serious challenges. The file similarity search schemes are widely used in digital forensics for efficient finding malicious files. And also, the file similarity search is very crucial and important to find duplicated information on storage system. Usually similarity search requires huge computation time and memory space, so all-to-all comparison is infeasible for large collection of files.
Well-known block hashing schemes are used for file similarity search[1,2]. Fixed-length chunking[3] lets files be divided into a number of fixed-sized blocks, and then

1 This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2011-0009358) (.2012R1A1A2044694) and (No. 2011-0029848).

ISSN: 2287-1233 ASTL

applies hash functions to extract a hash key of the blocks. Fixed-length chunking is very simple, light-weight scheme and is easier to implement than variable-length chunking[4]. But the drawback of fixed-length chunking is the data shifting problem. For instance, if some of data is inserted or deleted at the boundary of block, the result of hash value can be almost changed differently. So the similarity accuracy usually shows bad performance. In content-defined chunking or variable-length chunking, each block is partitioned using special hash values called anchor. This scheme can prevent the data shifting problem of fixed-length chunking which is a fast duplication approach. So variable-length chunking is more flexible than fixed-length chunking but requires more computation time to achieve good similarity accuracy. Piecewise hash[5] also widely used to detect similarity between files. This mechanism concatenates part of hash values and generates string values for each file. By adapting Locality-sensitive hashing (LSH) algorithm[6], it can find similar hash string. LSH is a method of performing probabilistic dimension reduction of high-dimensional data to hash the input items so that similar items are mapped to the same buckets with high probability.

In this paper, we propose a practical approach for minimizing metadata using file similarity information and hash key compression technique, we calls LwVLC. With a help of this mechanism, the required storage capacity can be hugely reduced. To provide more elaborated similarity search, we coupled piecewise hash algorithm and representative hash scheme. Here, we divide a file into several blocks and generate hash key for each blocks and make representative hash list. By concatenating each byte of representative hash value, we can generate hash string for each file. To find the level of file similarity, each hash string is compared to target files. We provide experimental evidence that our method gives significant improvement in running time over other methods for searching. We compare the effectiveness of LwVLC against fixed-length chunking (FLC) and variable-length chunking (VLC), those are existing well-known approaches.

2 System Design

In this section, we describe the architecture of LwVLC and explain how it works. The key idea of our work is three-folds: First, we adapt file similarity concept. File similarity concept is widely used for efficient file comparison. We can measure file similarity by comparing hash keys for each file blocks. The higher file similarity between two files means that less modified data blocks exist between two files. Second, we adapt variable-length chunking and extract representative hash list. In this approach, a file is divided into several variable-sized blocks with anchor. It gives more flexibility for inserting and deleting some data on a file. Finally, we adapt concatenation of hash keys to reduce metadata size and sorting scheme to improve searching time

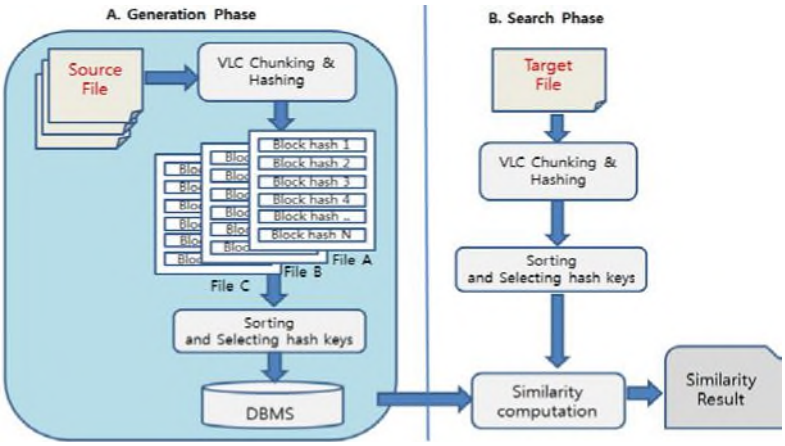

Fig. 1. The architecture of the proposed system

Figure 1 shows the overall architecture of the proposed system. It has several procedures to get the similarity result between a target file and a source file. We distinguish two phase, generation phase and search phase. The first is the generation phase that is devoted to generating and maintaining a collection of source files. DBMS with metadata of source files should be generated in order to compare a target file. The second is the search phase that is devoted to analyzing a target file using the material generated earlier. Figure 2 describes the procedure of generation phase. The main function is to extract file similarity metadata. Finally metadata will be stored in DBMS.

- Step 1 : Variable-length chunking and hashing
- Step 2 : Sort hash keys and select representative hashes
- Step 3 : Make string hash as metadata by concatenating 1byte LSB of each hash
- Step 4 : Store metadata in DBMS

In step 1, we have to scan each source file. Each source file is divided into variable-length sized blocks as finding anchor. Each block can have a hash value using SHA-1 hash function. In step 2, we have to sort metadata and choose only 100 maximum hash values as representative hashes. In this work, we made the representative hash list for all files. We extract 100 representative hash values from each file. The amount of additional for file similarity is not critical for metadata management. In step 3, we generate a hash string by concatenating 1byte LSB of each hash. Finally in step 4, all metadata of source files are stored in DBMS.

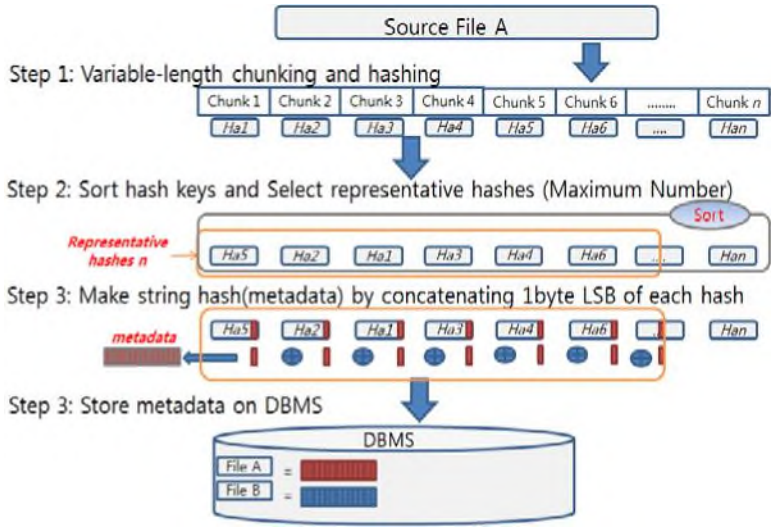

Fig. 2. Generation phase for building the reference collection with metadata

The algorithm finds anchor byte and seeks the current file position using *seek()* function. We can get a variable-length sized blocks from file stream. And then, we can generate new hash key using SHA-1 hash function with the byte of current position. The hash key is compared with previous hash keys. If the hash key is bigger than a hash key in hash array, we replace the new hash key with minimum value in hash array. This procedure repeats to the end of file and the output of this loop is the sorted hash list. We select representative hashes from sorted hash list. In our experiment, 100 hash values are selected to search file similarity. Finally, the metadata is formed by concatenating the one byte (8bits) least-significant bits of block hash value in order to reduce the required memory space and computation time.

3 Experiment Result

We experimented our scheme by comparing well-known similarity search approaches, fixed-length chunking and variable-length chunking scheme. We performed experiments on three data sets. We made the target files (size: 100MB) by patching a random data block to source file. In fixed-length chunking, chunk size is 4KB that is fixed, so total number of chunks is 25,600 because file size is 100MB. Each chunk has its own hash value. Therefore total number of hash values is also 25,600. In variable-length chunking, each chunk size is nearly 4KB, not fixed but variable. If there is no anchor byte in 4KB chunk, maximum chunk size can be up to 16KB. Therefore total number of chunks and total number of selected hash values are 64,000 up to 256,000 because also each chunk has its own hash value.

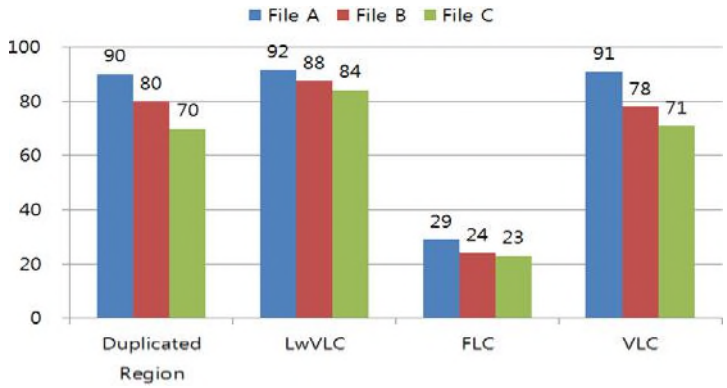

Fig. 3. Experiment results of file similarity in terms of accuracy

Figure 3 presents the file similarity result in terms of accuracy. As the figure shows, the similarity accuracy of FLC shows the worst case because of the data shifting problem. VLC shows best performance because it compares all the chunk data between files. In case of LwVLC, the performance result is comparable to VLC. As noted earlier, the purpose of LwVLC is to reduce processing time of similarity search, therefore, it extremely reduces metadata size. Accordingly, the accuracy result is more or less diminished. Figure 4 presents the result of execution time to search file similarity. As the figure 5 shows, LwVLC achieved the best performance, while VLC achieved the worst performance. LwVLC took just nearly 100 sec to search file similarity for file set (roughly, 100 files). The performance of LwVLC is better than FLC because FLC computes all hash values while LwVLC just compares one hash string between files. As a results, LwVLC achieved good similarity accuracy and reduced execution time for similarity search.

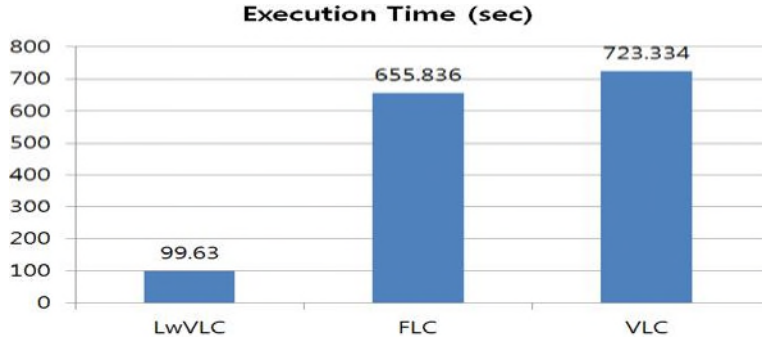

Fig. 4. Execution time for searching file similarity

4 Conclusion

In this paper, we propose a practical approach for minimizing metadata using file similarity information and hash key compression technique. The key idea is to reduce the metadata size by applying piecewise hash algorithm for each file. Furthermore, we

reduced hash values by adapting representative hash during variable-length chunking process. We compared the performance of this technique, variable-length chunking and fixed-length chunking. We showed that we can considerably improve the execution time to search file similarity with a good accuracy and reduce the metadata size.

References

1 Aronovich, L. and Asher, R. and Bachmat, E. and Bitner, H. and Hirsch, M. and Klein, S.T., "The design of a similarity based deduplication system", Proceedings of SYSTOR 2009, 2009.
2 Xia, W. and Jiang, H. and Feng, D. and Hua, Y., "Silo: a similarity-locality based near-exact deduplication scheme with low ram overhead and high throughput", in ATC, pp. 1-14, 2011.
3 Quinlan, S. and Dorward, S., "Venti: a new approach to archival storage", Proceedings of the FAST 2002 Conference on File and Storage Technologies, 2002.
4 Athicha M., Benjie C., and David M. "A Low-Bandwidth Network File System." In Proceedings of the Symposium on Operating Systems Principles (SOSP'01), pp. 174–187, 2001.
5 J. Kornblum, "Identifying Almost Identical Files Using Context Triggered Piecewise Hashing," Proc. 6th Ann. Digital Forensics Research Workshop Conf. (DFRWS 06), pp. S91–S97, 2006.
6 A. Andoni, P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in FOCS, IEEE Computer Society, pp. 459–468, 2006.

