Jan. 31
International Journal of advanced studies in Computer Science and Engineering
IJASCSE, Volume 4, Issue 1, 2015

REDUNDANT FILE FINDER, REMOVER IN MOBILE ENVIRONMENT THROUGH SHA-3 ALGORITHM

[1]MEERA.K [2]KRISHNA SANKAR.P [3]SRIRAM KUMAR.K
Department of Computer Science and Engineering
K.S.R Institute for Engineering and Technology

Abstract: Mobile environment provides storage as a main service. Data storage is a desired property when users outsource their data to be stored in a place irrespective of the locations. File systems are designed to control how files are stored and retrieved. Without knowing the context and semantics of file contents, file systems often contain duplicate copies and result in redundant consumptions of storage space and network bandwidth. It has been a complex and challenging issue for enterprises to seek deduplication technologies to reduce cost and increase the storage efficiency. To solve such problem, Hash values for files has been computed. The hash function competition to design a new cryptographic hash standard `SHA-3' is currently one of the well-known topics in cryptologic research, its outcome heavily depends on the public evaluation. Testing the finalists in the competition for a new SHA-3 standard shows generally fast, secure hashing algorithms with few collisions. Focus of computation is performed for duplicate knowledge removal. Hash computation is done by the method of comparing files initially and followed by SHA3 signature comparison. It helps to reclaim valuable disk space and improve data efficiency in mobile environment.

Keywords: Deduplication, Hash code, SHA3, Similarity Measure

Introduction

A hash function is any function that can be used to map digital data of arbitrary size to digital data of fixed size, with slight differences in input data producing very big differences in output data. The values returned by a hash function are called hash values, hash codes, hash sums, or simply hashes. One practical use is a data structure called a hash table, widely used in computer software for rapid data lookup. Hash functions accelerate table or database lookup by detecting duplicated records in a large file. They are also useful in cryptography. A cryptographic hash function allows one to easily verify that some input data matches a stored hash value, but makes it hard to reconstruct the data from the hash alone. When storing records in a large unsorted file, one may use a hash function to map each record to an index into a table *T*, and collect in each bucket *T*[*i*] a list of the numbers of all records with the same hash value *i*. Once the table is complete, any two duplicate records will end up in the same bucket. Duplicates can then be found by scanning every bucket *T*[*i*] which contains two or more members, fetching those records, and comparing them. With a table of appropriate size, this method is likely to be much faster than any alternative approach.A good hash function should map the expected inputs as evenly as possible over its output range. That is, every hash value in the output range should be generated with roughly the same probability. The reason for this requirement is the cost of hashing-based methods goes up sharply as the number of collisions pairs of inputs that are mapped to the same hash value increases. Basically, if some hash values are more likely to occur than others, a larger fraction of the lookup operations will have to search through a larger set of colliding table entries. A hash function that is used to search for similar data must be as continuous as possible; two inputs that differ by a little should be mapped to equal or nearly equal hash values.

Related Work

Files of similar type are taken to find similarity among them. Comparison of files is done based on the parameters like file size, storage occupied by it and contents of file. If files are same based on parameters, then it will be termed as deduplicated. Storage services commonly use deduplication, which eliminates redundant data by storing only a single copy of each file or block. Deduplication reduces the space and bandwidth requirements of data storage services, and is most effective when applied across multiple users, a common practice by storage offerings. It implies the privacy implications of cross-user deduplication. It demonstrates how deduplication can be used as a side channel which reveals information about the contents of files of other users. In a different scenario, deduplication can be used as a covert channel by which malicious software can communicate with its control center, regardless of any firewall settings at the attacked machine. Due to the high savings offered by cross-user deduplication, storage providers are unlikely to stop using this technology. Therefore, proposed simple mechanisms that enable cross-user deduplication while greatly reducing the risk of data leakage.

The fast growth of data volumes leads to an increased demand for online storage services, ranging from simple backup services to storage infrastructures. Remote backup services give users an online system for collecting, compressing, encrypting, and transferring data to a backup server that is provided by the hosting company.

Storage refers to scalable and elastic storage capabilities that are delivered as a service using Internet technologies with elastic provisioning that does not penalize users for changing their storage consumption without notice. The term data deduplication refers to techniques that store only a single copy of redundant data, and provide links to that copy instead of storing other actual copies of this data. With the transition of services from tape to disk, data deduplication has become a key component in the backup process. By storing and transmitting only a single copy of duplicate data, deduplication offers savings of both disk space and network bandwidth.

For vendors, it offers secondary cost savings in power and cooling achieved by reducing the number of disk spindles. According to recent statistics, deduplication is considered to be the most-impactful storage technology and it.

Initially, files of same type have been taken to be compared. Hash values are computed for those values based on MD5 algorithm in existing system and SHA-3 algorithm in proposed system. file detection systems is to provide useful information to users. The key idea of this system is to monitor file I/O periodically and check similar files in the storage system. Adapt similar hashing algorithm to check similarity between files on the storage system. First, focus on efficient hash algorithm which gives good performance without incurring heavy CPU processing and I/O requests. Second, interest in minimizing disk space by recommend similar files including duplicated files.

Computer system deals with lots of files to support various applications. As multimedia services are increasing, storage systems have to store large size data files including movie, virtual machine image, and CD image files. Many operating systems recently support deduplication-based file system to minimize disk storage capacity. However, this approach usually incurs heavy system

overhead. Another alternative method is to use a specific file manager application that supports elimination of duplicated files. However this approach also has difficulties from performance overhead.

Proposed Work

SHA-3 uses the sponge construction in which message blocks are XORed into a subset of the state, which is then transformed as a whole. Input files of similar type are gives as input for SHA-3 algorithm computation. Initially the state is assigned with zero. Padding is performed i.e. appending bits. Absorb the input into the state; that is, for each piece, XOR it into the state and then block permutation is applied.

The hash value for a given file is computed at a same rate as input. Then hash values of two files are compared to detect the duplication. If the hash values of similar type of files that had been taken are true, then one of the duplicate file is removed. Otherwise, it returns false.

SHA-3 can produce the hash values in desperate kind of bits like 224 bits, 256 bits, 384 bits and 512 bits. It computes hash values with XOR operation that takes 24 rounds. It is more anticipated more compared with MD-5 and SHA-1 algorithms.

SHA-3 has extendable output functions which is not likely in MD-5 and SHA-1 as well. The eloquent algorithm for computing hashes for a given output is SHA-3 to remove redundancy efficiently.

Figure3.1 :SHA-3 Hashing on a File

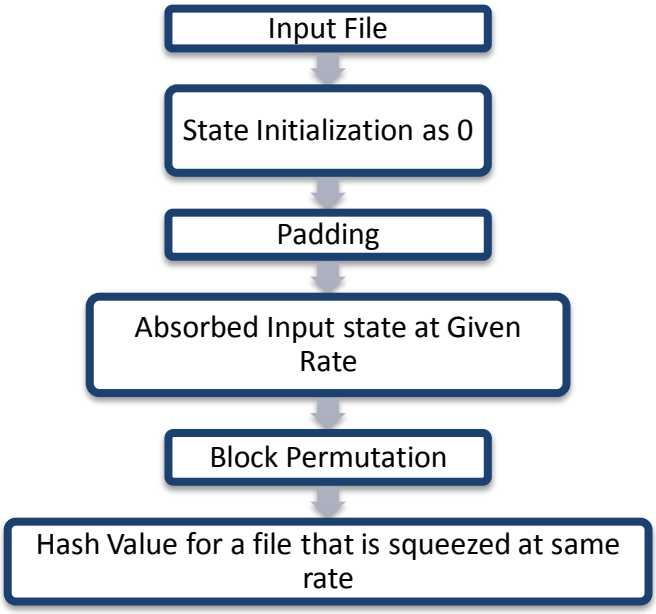
System Modules

4.1 Evaluating file properties

Initially the files that are taken as input is compared by parameters like size, memory occupied on the disk by each files and content of both files. If both files are same under these parameters, then it is loaded into the buffer to detect the similarity between the files and one of the files is removed by its hash values.

4.2 Generating hash values based on SHA-3

Hash values for files are manipulated under sponge construction which is defined by NIST. Padding of file is performed that is single 1 bit appended to the file followed by many zeros and length



is appended. Buffer is initialized to load the hexadecimal values of files.

4.3 File comparison

Hash values of various bits are obtained under SHA-3 algorithm. If hashes of two files are same, then it shows that same kind of file is copied at many places in an environment. Files are selected for comparison in Android Environment. One of the files is removed. Before produce the hash value, it checks content of file with another file. Hash value based on file content, is spawned.

4.4 Executing in android platform

Redundancy file comparison based on SHA-3 algorithm is implemented in android mobile environment. It checks the files of any type that are stored in phone memory and external storage as well. If it is phone memory, it requires a super user permission to detect and remove the files. Hash values of similar files like .txt, .jpeg, .avi is computed. It is removed if the hash values are same for the both files.

Result and discussion

The results are shown that memory consumption of files that are compared to remove the redundancy based on MD5 and SHA-3 hashes. Figure illustrates that size of files (as 5KB, 10 KB , ...) are taken and how it consumes storage on memory. Result of SHA-3 memory consumption is better than MD5. Memory occupied by MD5 is larger than SHA-3 algorithm. Thus SHA-3 greatly reduces the redundancy based on disk storage.

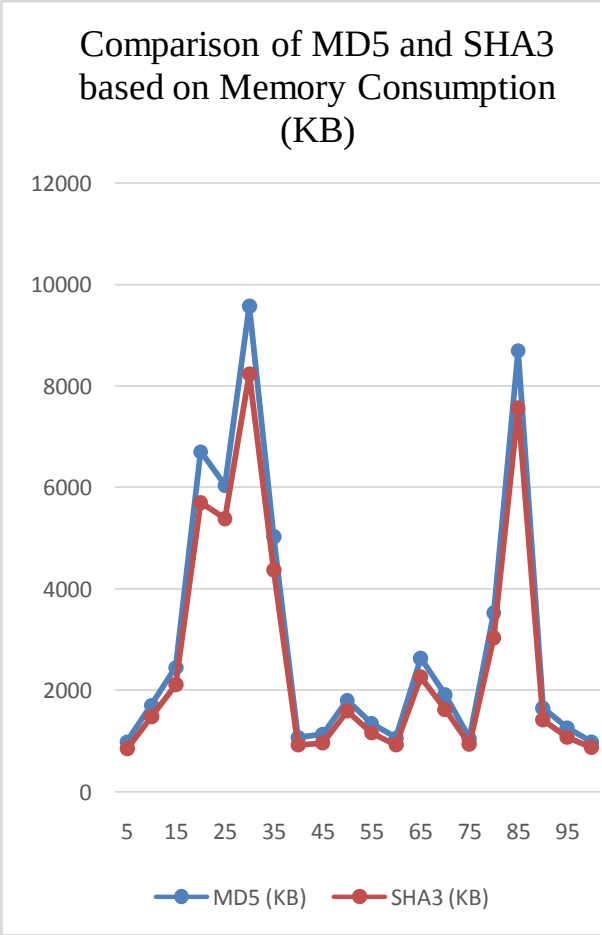


Figure 5.1: Comparison of md5 and SHA-3 based on memory consumption

The results are shown that time taken to compare and compute hash values for files to be taken based on MD5 and SHA-3 algorithm. Figure illustrates that files are of size (as 5KB, 10KB, 15KB, ..) taken and how much time it takes to compute hash value for files of various size. Time taken for SHA-3 algorithm to compute hash value is much lesser than MD5 algorithm. Thus SHA-3 algorithm removes similar files base on hash value with less time on mobile environment.

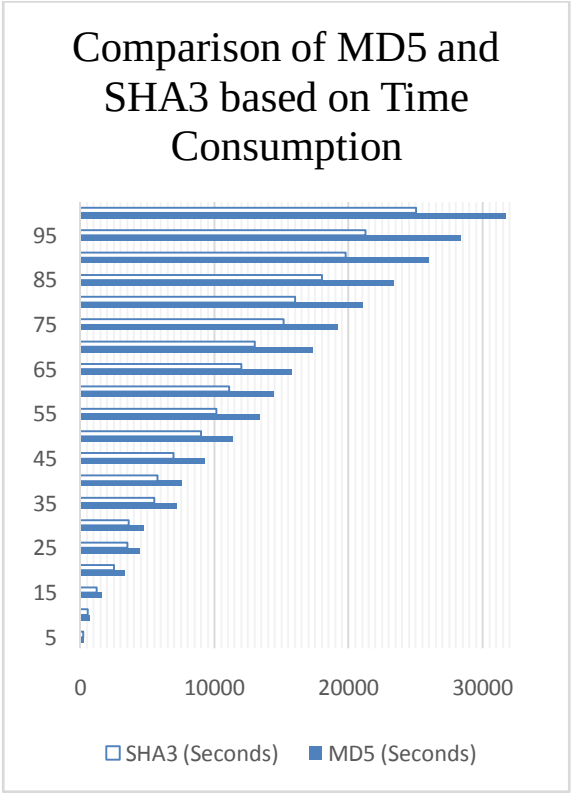


Figure5.2 : Comparison of md5 and sha-3 based on time

Conclusion

The file systems built into modern Operating Systems do not provide adequate support for managing file duplication. File duplication can be identified in detail with initial comparison of files, followed by MD5 algorithm in earlier and by SHA-3 algorithm in later. Based on MD5 and SHA-3, hash values for files have been generated. One of the redundant files is removed, if hash values of similar type files are same. The time taken to compute hash value by SHA-3 is much lesser than MD5. MD5 consumes more memory than SHA-3 algorithm. The performance of MD5 hash function is severely compromised in terms of memory consumption and time compared with SHA-3 algorithm. SHA-3 helps in retrieving valuable disk space and in improving the efficiency. SHA-3 is the best in identifying the redundant files in a mobile environment. The final SHA-3

candidates show much promise in terms of performance. SHA-3 hashing algorithm incorporated into an environment for detecting the redundant files and for removing it in mobile platform.

References

1. AL-Marakeby (2013) 'Analysis of MD5 Algorithm Safety against Hardware Implementation of Brute Force Attack', International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 9 pp.3332–3335.

2. Alia Arshad, Dur-e-Shahwarkundi and Arshad Aziz (2014), 'Compact Implementation of SHA3-512 on FPGA', Conference on Information Assurance and Cyber Security (CIACS).

3. Amar Jaffar and Christopher J. Martinez (2013), 'Detail Power Analysis of the SHA-3 Hashing Algorithm Candidates on Xilinx Spartan-3E',International Journal of Computer and Electrical Engineering, Vol. 5, No. 4 pp.410-413.

4. Danny Harnik, Benny Pinkas and Alexandra Shulman-Peleg (2010), 'Side channels in cloud services, the case of deduplication in cloud storage', pp. 1-7.

5. Dutch T. Meyer and William J. Bolosky (2012), 'A Study of Practical Deduplication', USENIX Association Berkeley, CA, USA ©2011, ISBN: 978-1-931971-82-9, pp.1-1.

6. FatmaKahri, BelgacemBouallegue, Mohsen Machhout and RachedTourki (2013), 'An FPGA implementation of the SHA-3: The BLAKE Hash Function', 10th International Multi-Conference on Systems, Signals & Devices (SSD) pp.1-5.

7. JaspreetKaur and Jasmeet Singh (2013), 'Survey on Efficient Audit Service to Ensure Data Integrity in Cloud Environment',Global Journal of Computer Science and Technology Software & Data Engineering, Volume 13 Issue 4 Version 1.0 pp.43-46.

8. Jin Kim, Sun-Jung Kim, and Young WoongKo (2014), 'Design and Implementation of File Monitoring Tools for Detecting Similar Files', 3rd International Conference on Computational Techniques and Artificial Intelligence pp.79-82.

9. KamleshkumarRaghuvanshi, PurnimaKhurana and PurnimaBindal (2014), 'Study and Comparative Analysis of Different Hash Algorithm',Journal of Engineering Computers & Applied Sciences (JECAS) ISSN No: 2319-5606 Volume 3, No.9 pp.1-3.

10. Mooseop Kim, DeokGyu Lee and JaecheolRyou (2013), 'Compact and unified hardware architecture for SHA-1and SHA-256 of trusted mobile computing', PersUbiquit Comput-2013 DOI 10.1007/s00779-012-0543-0 pp. 921–932.

11. Penny Pritzker and Patrick D. Gallagher (2014), 'SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions', Information Tech Laboratory National Institute of Standards and Technology pp.1-35.

12. Ram Krishna Dahal, JagdishBhatta and Tanka Nathhamala (2013), 'Performance Analysis of SHA-2 And SHA-3finalists', International Journal on Cryptography and Information Security (IJCIS), Vol.3,

13. Ruey-Kai Sheu, Shyan-Ming Yuan, Win-Tsung Lo and Chan-I Ku(2014)'Design and Implementation of File Deduplication Framework on HDFS', International Journal of Sensor Networks, Article ID 561340 pp.520-532.

14. ThulasimaniLakshmanan and MadheswaranMuthusamy (2012), 'A Novel Secure Hash Algorithm for Public Key Digital Signature Schemes', The International Journal of Information Technology, Vol. 9, No. 3 pp.262-267.