ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 10, October 2015

DupliCut: Redundant File Finder and Remover

Sumita Chandak[1], Abhishek Kadam[2], Bhagyashree Gawade[3], Jidnyesh Sankhe[4], Rohish Badkar[5]

Lecturer, Information Technology, Atharva College of Engineering, Mumbai, India[1]

Student, Information Technology, Atharva College of Engineering, Mumbai, India [2,3,4,5]

Abstract: Desktop environment provides storage as an essential service. Data storage is the aim when users outsource their data to be stored in a place irrespective of the locations. File systems are actually designed to control how files are stored and retrieved. Without knowing the context and semantics of the content, file systems often contain duplicate copies and result in redundant consumptions of storage space and network bandwidth. It has been a tedious and challenging issue for enterprises to seek reduplication technologies to reduce cost and increase the storage efficiency. In order to solve such problem, hash values for files have 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 largely 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 and problems. Focus of computation is performed for duplicate and redundant knowledge removal. Hash computation is done by the method of comparing files initially and followed by SHA3 signature comparison. Saves time and enhances accessibility. It helps you reclaim your valuable storage space and keep your data secure. It provides fast and efficient way to detect and remove duplicate file**s.** Therefore, ample memory can be saved by running DupliCut.

Keywords: Redundant data, DupliCut, SHA-3, Storage efficiency.

I. INTRODUCTION

As the voluminous of web documents increases on internet, it is a burden to search engines to provide the relevant information to the user query. In addition, more number of duplicates of documents also grows simultaneously on the web which increases the retrieval time and reduces the precision of the retrieved documents. Therefore to identify duplicate and near-duplicate web pages, researchers using the complexity algorithms rather using the classification algorithms [4].

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 [1]. 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 value [12].

II. LITERATURE REVIEWED

1. Meera K. Krishna Sankar P. and Shriram Kumar K. proposed to use theSHA-3 algorithm for comparison of files 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. It focusses on the following aspects: reducing the space and bandwidth requirements of data storage services, using SHA-3 algorithm for the generation of hash code of files, how deduplication can be used as a side channel which reveals information about the contents of files of other users. [1]

2. Suresh Subramanian, Sivaprakasam addressed storage space issues through Genetic Algorithm and Duplicate Web Documents Identification Function. Duplicate Web Documents Identification Function is used to improve relevance of retrieved documents by removing the duplicate records from the dataset. The paper focuses on detection and removal of duplicate web pages and nearly duplicated web pages from the dataset used for finding the

fitness function applied in Genetic Algorithm using rank based objective function. [4]

3. Nayana M S, Mrs Bindu A U presented a compact design of newly selected Secure Hash Algorithm (SHA-3) by dividing the basic Keccak architecture in to padder module and permutation module that reflects the sponge construction from which algorithm can be easily generated. The design techniques were proposed in the basic SHA-3 architecture in order to achieve better time performance. [9]

III. PROPOSED SYSTEM

The Keccak algorithm was the winner of the SHA-3 competition. The Keccak function is created using a number of sponge functions. The Keccak sponge function is made up of seven permutation functions of different bit lengths. The seven permutation functions are then used in XOR and composed of basic logic functions that can be reduced. The Keccak algorithm requires two different configurations for the 256-bit and 512-bit message digest sizes. The Keccak algorithm does not require the use of any RAM in the implementation [3]. 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. In [2], every single compression function of Keccak composed of 24 rounds. Every single compression function of Keccak composed of 24 rounds. The SHA 3 algorithm provides a good security for the data using the authentication format by generating hash code [2].

IV. SYSTEM IMPLEMENTATION

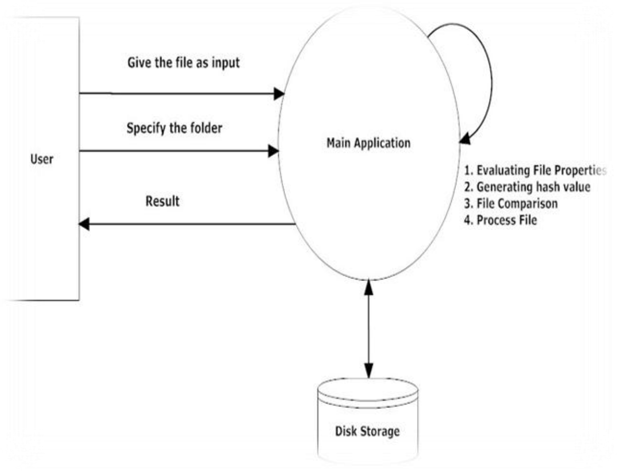


Fig. 1. Main application

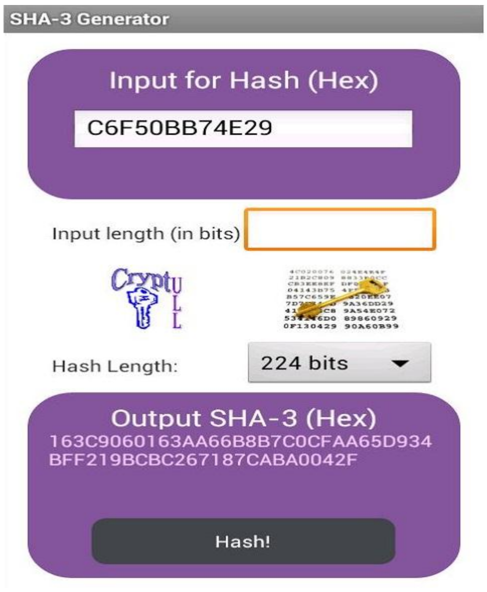
In Fig. 1, the main application working is described. The following paragraphs give detailed information about every step in duplicate file detection process.

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 [1].

GENERATING HASH VALUES BASED ON SHA3:

As shown in Fig 2, hash values for files are manipulated under sponge construction which is defined by NIST. Padding of the bits is compulsory with '0' and '1' first and last respectively until the resulting $\neq$ bit length which = 448 mod 512,and the last of bit length of the original message as 64-bit integer. The last bit length of the message which is already padded is 512N for a true integer N [11].


SHA-3 Generator

Input for Hash (Hex)

C6F50BB74E29

Input length (in bits)

Hash Length: 224 bits

Output SHA-3 (Hex)

163C9060163AA66B8B7C0CFAA65D934
BFF219BCBC267187CABA0042F

Fig. 2. SHA-3 Generator

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 windows 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 [12].The beauty of the new approach is simple, and the output list is free from redundant documents and the output list is having the more promising results [4].

EXECUTING IN .NET PLATFORM:

In Fig. 3, directions of how the process executes itself is given .File comparison is based on SHA-3(Keccak algorithm). It is implemented in dot net environment. It starts with checking for files in the given memory or space defined. Hash values of similar files like .txt, .jpeg, .avi, .mpeg is computed. If the hash values computed are same, then one of the files is removed according to given parameter.

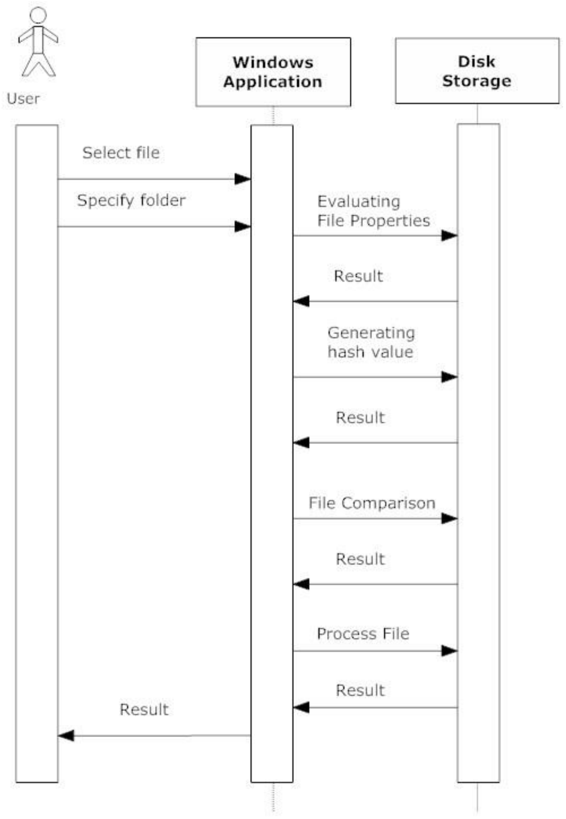


Fig. 3. Sequence of processes.

V. EXPECTED RESULTS

The DupliCut application, by using SHA 3 algorithm provides a good security for the data using the authentication format by generating hash code. Also the whole design has a simple hardware structure and fast running speed and can be widely used in digital signatures and 3DES key generation systems. [2]Also, the time complexity of SHA-3(Keccak) is less than its rivals like SHA-2 and MD5.

VI. 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, SHA-2, and SHA-1. SHA-1, SHA-2, MD5 consumes more memory than SHA-3 algorithm. The performance of SHA-1, SHA-2, 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 desktop 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.

It gives us great pleasure in presenting this project report titled: "DupliCut: Redundant File Finder and Remover." On this momentous occasion, we wish to express our immense gratitude to the range of people who provided invaluable support in the completion of this project. Their guidance and encouragement has helped in making this project a great success. We express our gratitude to our project guide Prof Sumita Chandak, who provided us with all the guidance and encouragement and making the lab available to us at any time. We also would like to deeply express our sincere gratitude to Project coordinators. We are eager and glad to express our gratitude to the Head of the Information Technology Dept. Prof Neelima Pathak, for her approval of this project. We are also thankful to her for providing us the needed assistance, detailed suggestions and also encouragement to do the project. We would like to deeply express our sincere gratitude to our respected principal Prof. Dr. Shrikant Kallurkar and the management of Atharva College of Engineering for providing such an ideal atmosphere to build up this project with well-equipped library with all the utmost necessary reference materials and up to date IT Laboratories. We are extremely thankful to all staff and the management of the college for providing us all the facilities and resources required.

REFERENCES

[1] Meera K. Krishna Sankar P. and Shriram Kumar nK(2015), 'Redundant file finder, remover in mobile environment through SHA-3 algorithm', Electronics and Communication Systems (ICECS), ISBN: 978-1-4799-7224-1, pp.1440-1447.
[2] 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] Suresh Subramanian, Sivaprakasam (2014),'Efficient Algorithm for Removing Duplicate Documents', International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-3, Issue-6.
[5] FatmaKahri, BelgacemBouallegue, MohsenMachhout 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.
[6] Jin Kim, Sun-Jung Kim, and Young WoongKo(2014), 'Design and Implementation of File Monitoring Tools for Detecting Similar Files', 3rd International Conference on ComputationalTechniques and Artificial Intelligence pp.79-82.
[7] KamleshkumarRaghuvanshi, PurnimaKhuranaand 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.
[8] Nayana M S, Mrs Bindu A U (2015), "Area-Efficient FPGA Implementation of Cryptographic SHA3-512", International Journal of Engineering Trends and Technology (IJETT), V21 (9), 455-460 March 2015. ISSN: 2231-5381.
[9] 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.
[10] Identifying and Filtering Near-Duplicate Documents Andrei Z. Broder? AltaVista Company, San Mateo, CA 94402, USA
[11] Piyush Gupta, Sandeep Kumar,'A ComparativeAnalysis of SHA and MD5 Algorithm' (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014, 4492-4495
[12] Krishna Sankar P. and Shriram KumarK(2015), 'Redundant File Finder, Remover In Mobile Environment Through SHA-3 Algorithm ',International Journal of advanced studies in Computer Science and Engineering IJASCSE, Volume 4, Issue 1, 2015.