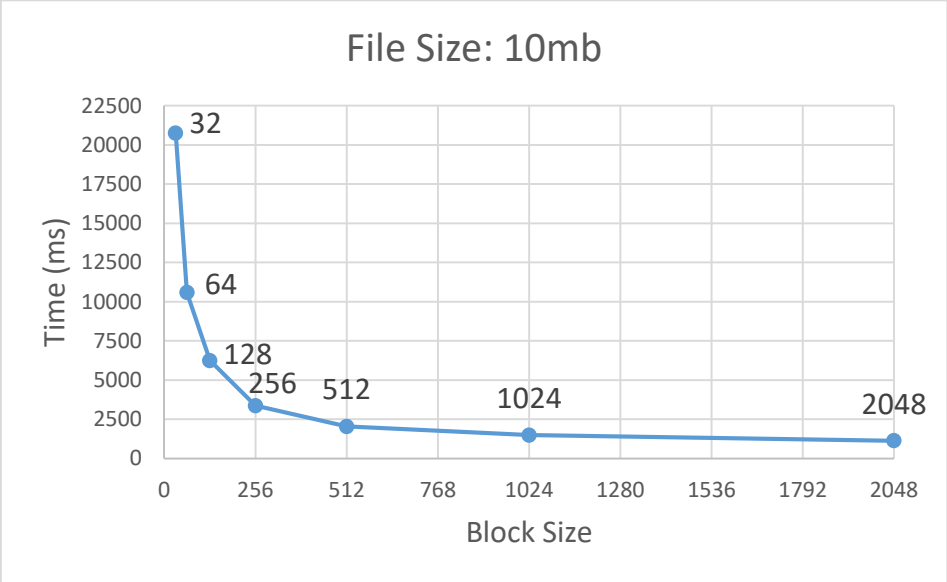
CS251 Fall 16 Project 3 Seungho Yang
PSO 10

*For the sumHash(), Megabyte.txt had the different Master Hash Value.
Hash Value on table: 53264
Computed Hash Value: 691840

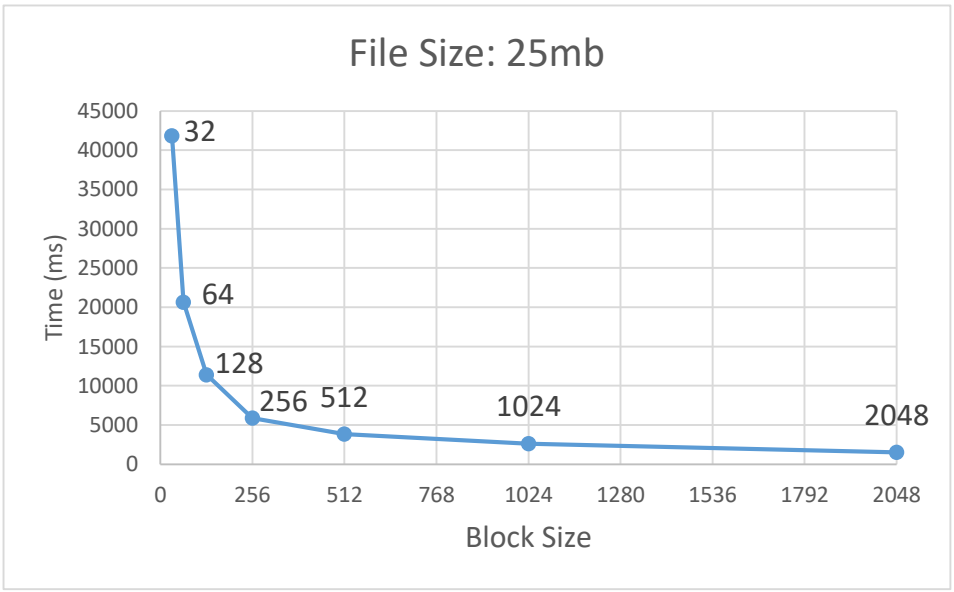
Experimental Analysis

Merkle Tree generation takes longer time as block size decreases. It takes more time to hash blocks of smaller size as it creates more nodes to hash. As block size doubles from each previous size, time almost halves.

File Size:	**10MB**
block size	Time (ms)
32	20753
64	10580
128	6245
256	3367
512	2043
1024	1486
2048	1131



File Size:	**25MB**
block size	Time (ms)
32	41843
64	20659
128	11418
256	5892
512	3843
1024	2634
2048	1538



Analysis Questions

Assuming that applying a hash function takes O(1) time. Answer each of the following questions in a clear and precise manner. Express time bounds in terms of m.

1. **Given a file that is divided into m blocks, what is the asymptotic worst-case time of building the Merkle tree from the m blocks?**

To generate a merkle tree, all of the nodes need to be generated with the hash function except the dummy node. When there are m blocks, then there are 2m nodes total. The first one will be the dummy node while the rest need to be generated using a hash function. It will take $2m - 1$ time to create each of those nodes. Asymptotic worst-case time of building a Merkle tree from m blocks is O(m).

2. **Given a leaf node u in the Merkle tree, what is the asymptotic worst-case time of determining the path siblings of leaf u?**

Starting from leaf u, the program must find the first sibling, move up to it's parent node and repeat the process until the last path sibling is either 2 or 3. Adding leaf u is out from the loop in my implementation and that only takes a constant time. The process loops while blockToTest > 1. Each loop divides blockToTest by 2 after adding the path sibling at the node of blockToTest index. If it's even numbered, odd numbered node is it's sibling and vice versa. It takes $\lfloor \log_2 2m \rfloor$ loops to find the path siblings. Therefore, asymptotic worst-cast time of determining path siblings of leaf u is O($\log_2 2m$)

3. **For a Merkle tree M containing m leaves, what is the asymptotic worst-case time of performing one Challenge-Response given a node in tree M?**

For a given node in Merkle tree, it takes at most $\lfloor \log 2m \rfloor$ loops to find the path siblings as stated in the problem above. Server will then pass the list of nodes to Client where it will run concatenateHash() method until it finds the master hash. Client generates parent node's hash value by running concatenateHash() on first two nodes in the list. For example, if leaf node was 16, list of path sibling nodes will be {16, 17, 9, 5, 3}. When you first concatenateHash() nodes 16 and 17, you will get the hash value of node 8. Executing concatenateHash() again with node 8 and 9 will get the hash value of 4 and so on until master hash is found. It takes $\lfloor \log_2 2m \rfloor$ hash functions until it finds the master hash to compare.

Challenge-Response will take a total of $2\lfloor \log_2 2m \rfloor$ time to verify if the program can find master hash with a given node. Asymptotic worst-cast time of determining path siblings of leaf u is O($\log_2 2m$).

4. **Let r be the root of a Merkle tree M (with m leaves) and let u its left child. Assume the server "lost" the hash values of nodes on the path from the left child of u to the leftmost leaf of M. For how many leaf nodes of M can the Challenge-Response be successful? You can assume that m, the number of leaves, is a power of 2.**

Challenge-Response will be successful for leaves only on the right child of the root of Merkle tree M. On the left half of the tree, the highest node with known hash value is u (or node 2). All the leaves on the left half of u are going to fail since the hash values of nodes on the path from left child of u to leftmost leaf of M are lost. Nodes from the right half of node u are also going to fail since hash value of the left child of u is lost as well.

With any of the leaves from the right child of the root (let's say *v*), Challenge-Response program will eventually reach to hash value of *v.* Running concatenateHash() on u and v will return the master hash value and the result will be successful. Number of nodes that will be successful will be m/2 since half of the m leaves are under the right child of the root.

5. Assume the Master Hash is obtained by concatenating the values at the leaves (strings obtained by hashing the file blocks) and hashing the obtained concatenated string. What are the advantages and disadvantages of this approach?

Since the program is hashing just the obtained concatenated string from all the hashed blocks, getting the Master Hash is faster than generating the whole merkle tree. Instead of running hash functions for all nodes of the tree, we just need to have one hash value. This will also make it easy for file verification. Instead of getting the entire file and checking that it's secure, we can generate master hash from two identical files and verify that the stored file is not changed or corrupted. This is much faster and allows verification of a file with minimum data transfer (as stated in the project descriptions).

However, if the stored file is corrupted or hash values from the file blocks become corrupted or lost, it will be much difficult to identify which portion of the file or hash values are causing the issue. If the stored file is corrupted and hash values of the merkle tree becomes corrupted or unavailable, it's easier to identify which portion of the tree is causing the issue. Like problem 4, based on finding which leaves fail Challenge-Response helps narrowing down which nodes or leaves are damaged. Since Master hash is generated from one large string, narrowing down the cause of the problem will be almost impossible.