To explore a more complex Merkle Tree, I generated eight random words and converted them into hashes using an online tool. These became the leaves for my Merkle Tree. Thereafter, I combined pairs of hashes to work my way up and create branches for the Merkle Tree until I was left with only the root hash – one which contained information about the all the hashes below it. I proceeded to change only one of the words in the leaves of the tree, and as expected, this led to a different hash in the leaf and all subsequent branches that followed from it. The root hash at completion was totally different than the one generated after the first attempt. I find interesting the different ways we can tackle problems that might arise in storing and transmitting data in this manner. What would happen if we only have an odd number of transactions? After all it seems unreasonable to assume there will ALWAYS be an even number of transactions made. A potential solution might be to double the hash that arises from the single, left-out leaf. To test how this would play out, I removed one of the original leaves and this time, instead of combining the hash from two different leaves, I simply doubled the hash that resulted from the lone leaf and worked my way up the Merkle Tree again. This method seems to work just fine – the algorithm that creates the hash does not care how the input string is generated and thus we are still able to create a root hash that can be used to verify data integrity. Thinking more about this though – although we are able to capture all the relevant information about the transactions, we have no way of saving the fact that we had to double the lone leaf. The root hash would be the same if the ‘fake’ hash had in fact been a real transaction and we would not be able to distinguish between the two. This may not be a problem in practice though since each transaction must be unique and therefore when using Merkle Trees to verify previous blocks of blockchain transactions – this method still serves our purpose.