483 HW1

Abdullah Atakan Guney - Sait Talha Nisanci
2013400123 - 2013400186

April 2018

In our solidity code there are 3 entry points for buying a ticket. We made those functions payable When buying any kind of ticket, full, half etc. users submit 3 hashes to these functions. A hash is the **keccak256** hashing of a random number chosen by the caller and the callers address, which will be used when revealing a number. When buying a ticket it is possible that the caller sent an excessive amount by an accident, in this case we sent back the excessive amount. In case of unsufficient funds we revert the call so that the money is sent back. After submission time for a round ends users send their random numbers to reveal function. When calculating the winners we are going to use **xor** of revealed numbers, Instead of storing all the reveal numbers in a collection like an array, we take the **xor** right away so that the gas price will be more even among the users. Otherwise at the last step we would need to iterate over the collection to calculate the winners and it would cost more to the last user who revealed their number. Also this saves from the storage.

After a reveal round ends, we determine the winners by usings the **xored** numbers. We make sure winners are different indexes by increasing the indexes when necessary and taking mod. When revealing users submit 3 numbers and **keccak256** hash is taken with the senders addresses and the reveal numbers and they are checked with the submitted numbers. Note that even though the submitted random number is correct if the sender address is different than the one that has bought the ticket then the function will revert. Users need to use the same address for submission and reveal rounds. After finding the winners the earnings are updated in a map so that they can be withdrawn in the future.

When withdrawing the amount to be withdrawn is sent to the function, if this value is more than the their earnings they will not withdraw anything. Withdrawing is a critical function. The callers money is decreased prior to sending it to prevent **DOA attacks**. If this was not done users could do recursive calls in client side and withdraw more than they have. Therefore withdraw method is secure against **DOA attacks**. Also note that when sending some amount from the contract, the result is checked and if it was not successful the function reverts otherwise some unintended results can occur.

As users can see the revealed numbers the last person can choose not to mine a block if they see that would result in them losing the lottery. This would

mean that they will lose the incentive which is 5 eth. It is clear that mining is more profitable in this case. So is is more incentive.

In all functions that are passed an array, array sizes are checked to prevent any undesired case. The order of submitted numbers and reveal numbers do not matter but to be able to successfully reveal random numbers one should reveal all the random numbers correctly. In case of any wrong number no eth will be refunded.

For buying tickets we chose to have three endpoints instead of one to eliminate one type parameter With three separate methods it is more clear to users which type of tickets they are buying and it is less error prone.

While testing the contract we have used truffle framework, as it takes a long time to mine 20K blocks we changed the round period to 20 blocks instead of 20K. In terms of functionality there is no difference as that is just a constant.