COMP4901W Homework 4 and 5

KIM, Jaehyeok (20550178)

The Hong Kong University of Science and Technology

Exercise 1

1. Proof that neither parties can cheat

As I suggested in the Homework 1, use of the hash function prevents either parties to cheat in my contract. Both Alice and Bob cannot guess the opponent's choice since it is hashed using `sha256` with a randomly chosen string `nonce` (*hiding property*). Moreover, the players cannot modify their choices after they submit their hashed choices (*collision-resistance propety*). Otherwise, `revealChoice()` will run into an error from

```
bytes32 hashed_reveal = sha256(bytes.concat(bytes(choice), bytes(nonce)));
require(hashed_choices[msg.sender] == hashed_reveal);
```

The `revealChoice()` can only be invoked after both players submitted their hashed choices, thus no player can look ahead the opponent's choice and then make their choice to win. Also, a player could resists to reveal his choice since he realized his defeat. This will not only waste his deposit but also the opponent's. To prevent this, my contract has a time limit for the revelation such that the players have to reveal their original choice in 1 hour since both players' submission of the hashed choices. If a player fails or resists to submit his choice in that period, the opponent will win the game and take the money.

2. Proof that no third-party can temper with the game

All functions of the contract check whether the address which invoked them is either Alice or Bob using `require(msg.sender == Alice || msg.sender == Bob)`. In other words, no third-party can successfully run the functions of the contract, thus there is no way for them to submit or change the players' choices. Moreover, unless the third-party has nearly or more than 50% of the hash rate, there is no way for them to censor the players' transactions for our contract.

Exercise 2

2. What if highest bidder refuses to pay?

If so, the corresponding auction has to be cancelled and restarted, and this would have revealed the other people's bids; losing the point of doing sealed-bidding auction. To ensure highest bidder to make the payment, my contract requires participants to pay the bid when revealing their original bid. Also, it receives a high deposit (e.g. 50 ETH $\approx$ 151k USD) to encourage participants to reveal their original bids. As the bids should remain hidden during the bidding period, the participants are first sharing their hashed bids. After the bidding period, each participant will reveal and pay their original bids to the contract. If a original bid is the new current highest, the contract pays back the previous winner and keeps the new current highest money. If their original bid is not the current highest, the contract returns the paid amount of money. In this mechanism, all non-winners will get returned their paid bids. Also, the highest bidded participant has already paid to the contract so it would be impossible for winner not to pay. People who unrevealed their original bids will not be able to get back the deposit, thus encouraging people to reveal.

3. How much gas do the contract use?

Since all participants are rational, they will only invoke really required functions without any useless or meaningless actions wasting their money for gas. Assuming that the contract instance has been created by seller, all participants will invoke the functions in the same order as follows: `bid()` $\rightarrow$ `revealBid()` $\rightarrow$ `getRefund()`.

`bid()`: checks the time bound condition, the amount of deposit, and whether the user has bidded already. Every invokation takes around 68829 gas and this gives an upper bound of 70000 gas for this function.

`revealBid()`: checks the time bound condition, whether the user has bidded, whether has revealed the original bids, and whether the paying amount is equal to the original biding price. Then, it checks whether provided original bid and the nonce are corresponding to the previously submitted hashed bid. And the winner decision precedure takes in place mentioned in 2.2. Every invokation takes around 74314 gas and this gives an upper bound of 75000 gas.

`getRefund()`: checks whether revealing period has end, if the sender has got refunded before, and if he has properly revealed his original bid. If all conditions pass, it sets the sender has received the refunds and pay the sender the deposit. Every invokation takes around 55217 gas and this gives an upper bound of 57000 gas.

Overall, the upper bound of gas fee used for each participant through the auction is 70000 + 75000 + 57000 = 202000 gas.