CrowdPricer

Smart Contract Assignment Report

Xiaoyao Qian (qian13)

Overview:

The ultimate goal of this smart contract is to provide a decentralized way to introduce external knowledges(e.g. Exchange rate, weather). In the meanwhile, the Ethereum blockchain does not provide means to know external knowledges. The only way to introduce external knowledge into a contract is to have a centralized entity to feed the data into the contract. The contract could only choose to believe the data feed provider won't do evil. However, if we leverage some game theory and incentives, the situation can change and we can have a decentralized way to know external knowledge.

Intuition:

The intuition behind our project is that, when an mutually-untrusted party is asked about the price of an item, she will get compensated if her answer gets closer to the median of all answers collected. If her answer is way off, she will get penalized economically. With this setting, everyone in the group will try to guess what the others might answer. The best option in this case is to actually do some research and provide an answer that is closest to the real answer(external knowledge). Eventually, the median value of all answers is somehow a trustworthy answer.

Contract Design:

As a starting step towards that ultimate goal, the project now enables any user to start the contract to ask for a group of others to price a specific item. When the contract starts, the initiator will provide the following:

- itemDescription: the description of the item she wants others to give price for.
- bounty: the amount of ether the initiator will compensate the group.
- tolerance: how much difference do the initiator allows from the median answer.
- minNumProposals: the minimum number of proposals/answers the initiator needs before the median is revealed and compensations are allocated.
- admissionDeposit: the amount of ether a participant proposer must deposit to the contract before she provides the answer. This variable is designed to decrease the possibility that one person create a lot of shadow accounts to increase his answer weight. No matter if the proposer ends up being compensated or penalized, this admissionDeposit amount will always be returned.

- confidenceBet: the amount of ether a participant proposer must deposit as bet to the contract. If her answer is not within the tolerance range, the bet will not be refunded and will be used as compensation to others.
- initiatorDeposit: the amount of ether the initiator must deposit to the contract. initiatorDeposit > minNumProposals * (admissionDeposit + confidenceBet). This is designed to increase the cost if the initiator do evil and wants to trap proposers' money in the contract. In that case, the initiator has to trap her own ether of the same amount as well.
- A maxNumProposals variable will be calculated based on initiatorDeposit / (_admissionDeposit + _confidenceBet). Once maxNumProposals is reached, no more proposals are accepted. If proposals are still accepted beyond this point, then the initiator could trap a larger amount of others' money than her own.

When a proposer wants to propose a price, she needs to send with an ether amount > admissionDeposit + confidenceBet before her proposal will be considered. The excessive amount will be returned back immediately.

The initiator can abort if no one has yet proposed. The contract will return initiator's deposit and bounty back on abort.

When there're less than minNumProposals, the initiator can choose to reveal at this point. Decisions on compensation and penalty will be made. The proposers can choose to reveal only when the minNumProposals are reached. Reveal can only be called once through the entire lifecycle of the contract. During the reveal, all proposals will be sorted, and based on the tolerance, decisions on compensation and penalty are made accordingly. Then each participant in this contract(including the initiator) can call claimRefund to collect their funds in the contract. Each fund cannot be collected more than once.

More details can be found in the code.

NOTE: I use the ethereum-testrpc(https://github.com/ethereumjs/testrpc) and truffle(https://github.com/ConsenSys/truffle) to develop this project. When truffle console is connected, you can interact with the contract. Example commands looks like:

```
a0="0xe76ecb9e897190c3b1617dbb1705d95445e84505"
a1="0x212d78e23d62260fe5bb2d777c010abb9f915025"
a2="0x99feee8d4201caa859fc4edbcd964ec91c387353"
a3="0xccda43ec2cf311b1480ec732a8408f470e345afb"
a4="0x6cdf950fe8d244795bc1c5a36253c71f188901ed"
a5="0x838a3dcc42110a7833d7beee4c79d093680cf724"
a6="0x966088b1cc85d4acb994ffac06d454f3f561ac9f"
a7="0x38c1d1201fa8ef32e2d2c901383fbe6cae4cdb43"
a8="0xc17bc6209e102bf486c8937229f992fab3cd5273"
```

```
a9="0xb3d3ae73456aca79e93b5cf8f3210b79750b8f02"

cp = CrowdPricer.deployed()

cp.propose(10, {from: a1, value: 110})
cp.propose(12, {from: a2, value: 110})
cp.propose(13, {from: a3, value: 110})
cp.propose(16, {from: a4, value: 110})
cp.reveal({from: a0})
cp.finalPrice()
...
```

Case Analysis:

Typical Case:

The initiator and proposers behave honestly. The initiator specify the settings, proposers propose their price with deposit and bet. When reveal can be called, anyone can call reveal and claim their refunds.

When the Initiator is Evil:

The initiator can be evil if she wants to trap proposers’ money in the contract by not revealing. This is costful in our project. The initiator has to place a deposit >= all money that proposers put in the game. If she tries to trap others, her own fund will also be trapped.

When the proposer is Evil:

The proposer can be evil if she wants to provide inaccurate proposal, but it is likely to be off the median answer, and her bet will be collected to compensate others.
The proposer can also try to reveal frequently before other proposers join, but our contract forbid proposers to reveal if minNumProposals is reached.