Blockchain based Bidding System

Bavya Balakrishnan
192IT003

Introduction

- An auction is a process participants make larger and larger bids for the commodity until it gets sold to the person who is ready to pay the most.
- E-Auction integrates the internet and ordinary auction to significantly reduce the transaction and transportation cost.
- Open auction is a Real time bidding where anyone can participate and they can bid against each other for buying the product. Finally, the person with highest bid wins the product.
- Participants are refunded if they do not win the auction.
- In the end the bid amount of winner is transferred to beneficiary.
- **Traditional Bidding system**
- Auctioneer (or beneficiary) is the person who conducts auction and announce if the good is sold to the winner.
- Posting the product, checking the highest bid and declaring the winner are handled by third party and most of the time it is centralised e.g. eBay and yahoo bidding system
- Privacy leakage when the personal data and transaction records are stored in a centralised database.
- Blockchain is a secure distributed ledger as nobody can interrogate the cryptographically secured blocks.
- The rules and policies for bidding can be defined in smart contract.
- As it is a distributed system, transaction cost is reduced significantly.

Literature Survey

Traditional Bidding system

Authors	Methodology	Advantages	Disadvantages
Ilichetty S Chandrashekar, Y Narahari, Charles H Rosa, Devadatta M Kulkarni, Jeffrey D Tew, and Pankaj Dayama[2007]	Instead of sending the actual bid the bidder sends a modified version of their bid amount hashed with a secret key.	no competition or pressure towards the end of auction. Once the bidding time is over The valid highest bid will be considered as the winner	it cannot always ensure if the bid price has been leaked by a third party or an adversary before the reveal time starts.
K. Omote and A. Miyaji. [2001]	English auction protocol where bids are registered on a bulletin board. It consists of 2 authorities. One is a Registration manager who registers the bidders and second one is an Auction manager who records bids.	Anonymity, Traceability, No framing, Unforgeability, Fairness, Verifiability and Efficiency, easy revocation of bidders	Registration manager and Auction manager are centralized entities

Literature Survey

Blockchain for E-Auction

Authors	Methodology	Advantages	Disadvantages
Yi-Hui Chen Shih-Hsin Chen Iuon-Chang Lin [2018]	With decentralized access structure, all bidders can bid the product by calling the open contract's trading contract without intermediate brokers.	E-auction mechanism based on blockchain to ensure electronic seals confidentiality, non-repudiation, and unchangeability	Bidders may call the wrong contract function. Lack of authentication
E.-O. Blass and F. Kerschbaum [2018]	New secure two-party comparison mechanism executed between any pair of bids in parallel. Using zero-knowledge proofs, Strain broadcasts the results of comparisons on the blockchain in a way that all participants of bidding can verify each outcome.	Guarantees bid confidentiality against fully-malicious bidders Low latency	Strain's latency is not asymptotically optimal To reach consensus, blockchain miners generally require access to all contract input data.
Alex Atallah Devin Finzer [2018]	It is a peer-to-peer marketplace where goods such as gaming items, digital art, and other goods backed by a blockchain can be bought and sold.	Bidders can bid with any amount, not necessarily higher than the highest bid	This protocol does not guarantee non-cancellation of bids, neither that the highest bidder wins.

Outcome of Literature Survey

- Most of the traditional bidding systems are based on Centralised approach which requires high transaction cost.
- It never guarantees whether the third-party can be trusted
- Blockchain technology with low transaction cost,enhanced security and privacy is used to develop the smart contract for public bid and blind bid
- Researchers are now focused on improving scalability, reducing latency and handling authentication of bidders.

Problem statement

To develop a smart contract based Open Auction(both Simple and Blind Auctions) using Ethereum Blockchain

Objectives

- Design and implement a simple E-Auction in Ethereum platform
- Design and implement a Blinded E-Auction in Ethereum platform
- Create a UI for building a Decentralized Simple Auction App with Ethereum
- Evaluate the framework using different test cases

Methodology

- We create smart contract for both Simple auction and Blind auction in Solidity
- A Solidity contract can act as an agreement between a buyer and a seller when selling an item remotely in E-Auction.
- We use web based Remix tool for compiling solidity smart contract.

Simple Auction

- During the predetermined bidding period everyone can send their bidding amount.
- Smart contract will update the highest bidder and highest bid according to the public bids it receives.
- At any point of time if highest bid is raised then the previous highest bidder must be refunded
- Those who do not raise any highest bid in the auction will be reverted
- Once the bidding period is over beneficiary or seller manually call the contract to receive his money
- function bid is made payable to enable the function to receive ether
- It is safer to let the recipient (previously highest bidder) to withdraw their money themselves. The function withdraw is used for that
- Once we compile the contract we instantiate it using the Ethereum javascript library
- We can take inputs from the UI required for invoking the smart contract and then call the corresponding function
- The UI is created using HTML and CSS

Algorithm 1 bid payable

```
INPUT: the value you sent along becomes your bid
REQUIRE: auctionStart, biddingTime,highestBid
if currentTime > auctionStart + biddingTime then
  revert the call
end if
if msg.value <= highestBid then
  send the money back and exit
end if
if highestBidder! = 0 then
  pendingReturns[highestBidder]+ = highestBid
end if
highestBidder = msg.sender
highestBid = msg.value
create event HighestBidIncreased(msg.sender, msg.value)
```

Algorithm 2 withdraw

```
OUTPUT: bool success
REQUIRE: auctionStart, biddingTime,highestBid
amount = pendingReturns[msg.sender]
if amount > 0 then
  pendingReturns[msg.sender] = 0
  if !msg.sender.send(amount) then
    return false
  end if
end if
return true
```

Blind Auction

- Bidder will be sending the hashed version of actual bid with a secret key blindedBid=keccak256(abi.encodePacked(value, fake, secret))
- In addition to biddingTime we must predefine the revealTime
- Bidding has to be performed before biddingTime expires
- The bid is considered as valid if the ether you send along with the bid transaction >= ‘value’ and fake is set to false
- smart contract restricts the refunds to only the bidder who can reveal its blinded bid during reveal period
- Refunds will be available for all topped bids and invalid bids that were blinded properly
- Once the revealEnd is reached then anyone can manually end the auction and the winning bid amount will be send to the beneficiary

Algorithm 3 bid payable-Blind Auction

```
INPUT: _blindedBid
REQUIRE: biddingEnd, struct Bid with blindedbid and deposit,
mapping bids from address to Bid[]
if biddingEnd < currentTime then
   EXIT
end if
bids[msg.sender].push(Bid(_blindedBid, msg.value))
```

Algorithm 4 reveal-Blind Auction

```
INPUT: values[],fake[],secret[]
REQUIRE: biddingEnd, struct Bid with blindedbid and deposit,
mapping bids from address to Bid[],length = bids[msg.sender].length,refund = 0
if biddingEnd > currentTimeor revealEnd < currentTime then
  EXIT
end if
for bidToCheck in bids[msg.sender] do
  if bidToCheck.blindedBid != keccak256(abi.encodePacked(value, fake, secret)) then
    continue
  end if
  refund+ = bidToCheck.deposit
  if !fake and bidToCheck.deposit ≥ value then
    if value > highestBid then
      refund -= value, pendingReturns[highestBidder] += highestBid
      highestBid = value, highestBidder = bidder
    end if
  end if
  bidToCheck.blindedBid = bytes32(0);
end for
msg.sender.transfer(refund)
```

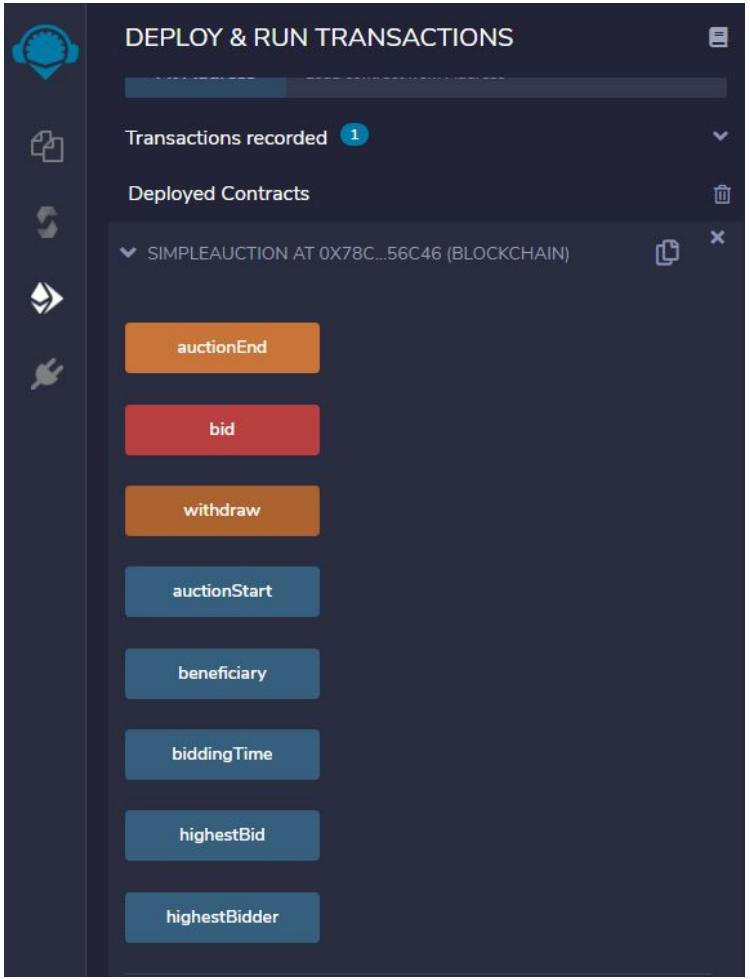
Implementation

Simple Auction

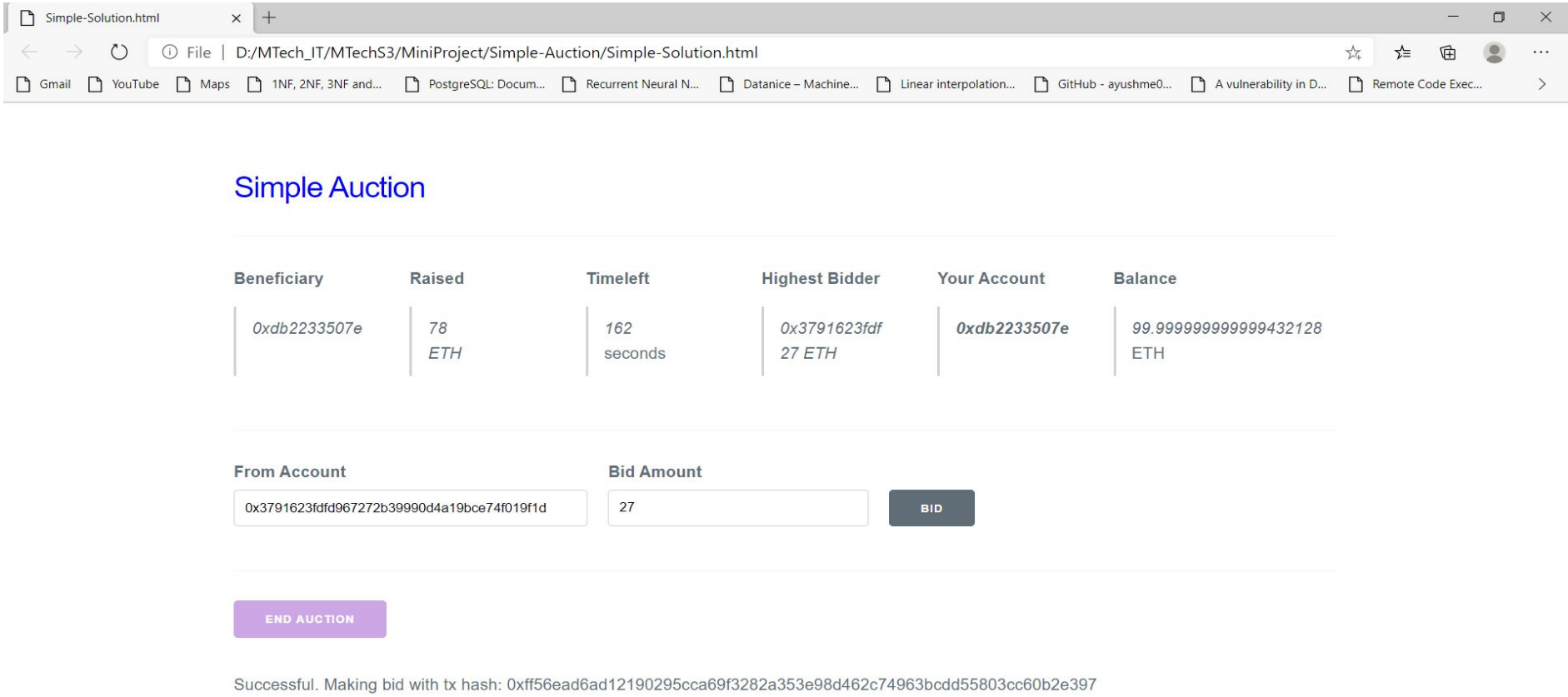
- In order to develop **Decentralized Simple Auction App** with Ethereum we created a UI.
- We can call the functions of simple auction through the UI and smart contract updates the highestBid, highestBidder, Account balances on UI.
- We set the biddingTime and Beneficiary address in the Javascript part of HTML.
- To test the framework of simple auction we use ganache-cli which is a personal blockchain for Ethereum development you can use to deploy contracts, develop your applications, and run tests to evaluate it
- We included 1) ethereumjs-testrpc: to simulate full client behaviour 2)ethjs:designed for building light-weight dApps to act as simple javascript interface for Ethereum nodes and clients
- We start the Ganache-cli in local machine and connect to the global blockchain by selecting listening port 8545 in Web 3 provider the Environment of Remix
- Exceptions are thrown when any of the following happens
    - If the bid is less than or equal to highest bid
    - If the bid is placed after bidding period is over
    - When the ENDAUCTION button is pressed before bidding period is over
    - ENDAUCTION is pressed more than once after bidding
- Once the bidding period is over and ENDAUCTION button is pressed then the highest bid is sent to the beneficiary

```
Ganache CLI v6.12.1 (ganache-core: 2.13.1)

Available Accounts
==================
(0) 0x430A8e4606F2546Af7C881A2D0aD68FD51aEbD14 (100 ETH)
(1) 0x354643f65D7a8b0e877e8448e002D58Ed2C2dbE7 (100 ETH)
(2) 0x2f710509A9f09b554023aa8d89c508eCc035a856 (100 ETH)
(3) 0x1e6150ab34b7550D0A8e7aCb1E4D991a1e94Ae3c (100 ETH)
(4) 0xEA436D0121Be8b32E31C7F14834167acaCeeF970 (100 ETH)
(5) 0x588bdfB2b222d6562f7cf97429a52e49113Fa685 (100 ETH)
(6) 0x25ed1037d60c1bB083295B8E995A320DfB621975 (100 ETH)
(7) 0x9FfcB126856a9492A1944274695A2E7ed58cF03c (100 ETH)
(8) 0xFd3A6C36f48d74E2df1261f40e0339B4Bc13883E (100 ETH)
(9) 0x322f53F0d54dCfE2Eda65e738C69fbeF33a7409e (100 ETH)

Private Keys
==================
(0) 0xe8c4f906ad8f2ca9d49e39339df48a67ee78ecda038b99d400314d1abef21d2a
(1) 0xce4b98bbb038ce20a8e018a6de937b4e0a260fa26a294a9cd1d97743a05197e4
(2) 0xd95a61585e70758643f12e20fc86059f8698633d36d3d939f71a1f3421b73821
(3) 0xe9a4704186b9ffdd1e08ae446a42c5393867a1a84b49df3bfcc2fbb5d59085ae
(4) 0xddb3202d9cfb62435c0b1dbb9ad395c38f78d7b24520af039938f70068062153
(5) 0xd4667569afda9a16c12275e34c15fd1aa278b5af463c753c8e00c132a2f133e0
(6) 0xbb17edf60966366e8b8c25f21762b64a5a18beb16dc489980bc25b834cc09eff
(7) 0x2db3c1cfca0256def2784f4d26f07e15df7b5f7fa27985751fa7a86848608d4d
(8) 0x80d476ed68cbdb34b4b9b466d3168e33f14740a6f0b789418f548f432c472b4b
(9) 0xe6b4ea92544f52fcf51a3624659abcced31013476e85d92f12d026a59f1b5564

HD Wallet
==================
Mnemonic:      always tiger lucky hawk auction fragile foil speed jar crawl loyal skin
Base HD Path:  m/44'/60'/0'/0/{account_index}

Gas Price
==================
20000000000

Gas Limit
==================
6721975

Call Gas Limit
==================
9007199254740991

Listening on 127.0.0.1:8545
>
```

Ganache cli listening on port 8545



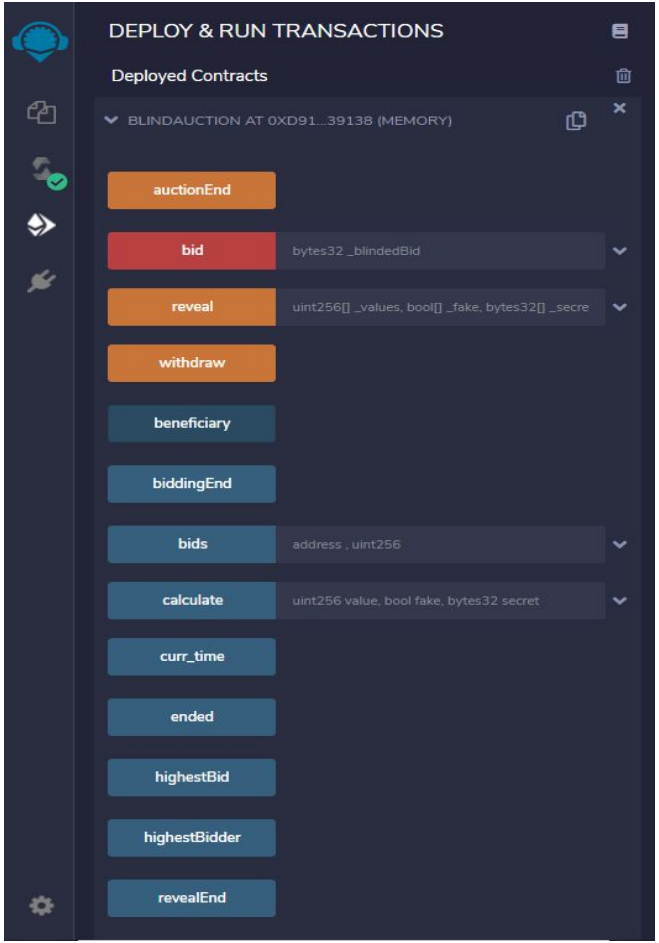
Simple Auction smart contract



- Based on the bidding time we fixed, the UI shows the Time left
- Information about the highest bidder and bid amount is also visible to all
- Raised field in the UI shows the value in ether owned by smart contract. This will be the sum of topped bids. Once the winning bid is transferred to the beneficiary, smart contract will be left with the bids of all topped bids. They must be withdrawn by bidders themselves

Blind Auction

- **Blind auction** smart contract is deployed and run on JavaScript VM
- The bidders place the blinded version of their bid till the reveal time starts.
- Once the bidding period is over the revealing starts.
- All the bidders who successfully revealed their bid but couldn't win are refunded immediately.
- Topped bidders who didn't win have to call the withdraw function manually to get their money.
- Exceptions raise in following situations.
    - Bid is placed after bidding period is over
    - Reveal is called after reveal phase ends or before bidding ends
    - Bidders couldn't reveal their bid correctly after auction
    - Someone tries to end the auction before reveal time is over
    - Someone tries to end the auction multiple times to invoke the transfer of ether
- The Blind auction smart contract is evaluated using a private blockchain network also



Blind Auction smart contract

DEPLOY & RUN TRANSACTIONS

Web3 Provider

Custom (2113) network

ACCOUNT

0xEf8...283F4 (16524.616962246 ether)

GAS LIMIT

3000000

VALUE

0 wei

CONTRACT

BlindAuction - browser/blindAuction.sol

Deploy 200,200,0xEf81e07f73527653DfAC91DC2A

Publish to IPFS

OR

At Address Load contract from Address

Transactions recorded 2

Deployed Contracts

BLINDAUCTION AT 0X525...03CA5 (BLOCKCHAIN)

Home blindAuction.sol 2 tabs

```
pragma solidity >0.4.23 <0.7.0;

contract BlindAuction {
    struct Bid {
        bytes32 blindedBid;
        uint deposit;
    }

    address payable public beneficiary;
    uint public biddingEnd;
    uint public revealEnd;
    bool public ended=false;

    mapping(address => Bid[]) public bids;

    address public highestBidder;
    uint public highestBid;
    mapping(address => uint) pendingReturns;

    event AuctionEnded(address winner, uint highestBid);
    modifier onlyBefore(uint _time) { require(now < _time); _; }
    modifier onlyAfter(uint _time) { require(now > _time); _; }
    constructor(
        uint _biddingTime,
        uint _revealTime,
        address payable _beneficiary //You can use .transfer(..) and .send(..) on address payable
    ) public {
```

0 listen on network Search with transaction hash or address

creation of BlindAuction errored: Returned error: authentication needed: password or unlock

creation of BlindAuction pending...

[block:8186 txIndex:0] from: 0xEf8...283F4 to: BlindAuction.(constructor) value: 0 wei data: 0x608...283f4 logs: 0 hash: 0x34c...a9e65 Debug

Blind Auction deployed on Private Blockchain

DEPLOY & RUN TRANSACTIONS

BLINDAUCTION AT 0XD91...39138 (MEMORY)

auctionEnd

bid 0x444b7ae584bcd1c8e7fffd44ee520ec

reveal [25],[false],["0x313200000000000000(

withdraw

beneficiary

biddingEnd

bids address , uint256

calculate 25,false,0x313200000000000000000

0: bytes32: 0x444b7ae584bcd1c8e7fffd44ee520edd520356d684ad26129ee4a624a1cd81d4

curr_time

ended

highestBid

Home | AccessControl_contract.sol | Register_contract.sol | blindAuction.sol | 4 tabs

```
6       uint deposit;
7   }
8
9   address payable public beneficiary;
10  uint public biddingEnd;
11  uint public revealEnd;
12  bool public ended=false;
13
14  mapping(address => Bid[]) public bids;
15
16  address public highestBidder;
17  uint public highestBid;
18  mapping(address => uint) pendingReturns;
19
20  event AuctionEnded(address winner, uint highestBid);
21  modifier onlyBefore(uint _time) { require(now < _time); _; }
22  modifier onlyAfter(uint _time) { require(now > _time); _; }
23  constructor(
24      uint _biddingTime,
25      uint _revealTime,
26      address payable _beneficiary //You can use .transfer(..) and .send(..) on address payable
27  ) public {
28      beneficiary = _beneficiary;
29      biddingEnd = now + _biddingTime;
30      revealEnd = biddingEnd + _revealTime;
31  }
32  function bid(bytes32 _blindedBid)
33      public
34      payable
35      onlyBefore(biddingEnd)
36  {
37      bids[msg.sender].push(Bid({
38          blindedBid: _blindedBid,
```

bytes32 _blindedBid 1 reference(s)

0 listen on network Search with transaction hash or address

[vm] from: 0xab8...35cb2 to: BlindAuction.reveal(uint256[],bool[],bytes32[]) 0xd91...39138 value: 0 wei data: 0x900...00000 logs: 0 hash: 0xacb...0d9f2 Debug

transact to BlindAuction.reveal errored: VM error: revert. revert The transaction has been reverted to the initial state. Note: The called function should be payable if you send value and the value you send should be less than your current balance. Debug the transaction to get more information.

Reverted transaction while performing reveal before bidding ends

DEPLOY & RUN TRANSACTIONS

ACCOUNT

0x4B2...C02db (100 ether)

GAS LIMIT

3000000

VALUE

25 ether

CONTRACT

BlindAuction - browser/blindAuction.sol

Deploy 200,200,0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

Publish to IPFS

OR

At Address Load contract from Address

Transactions recorded 2

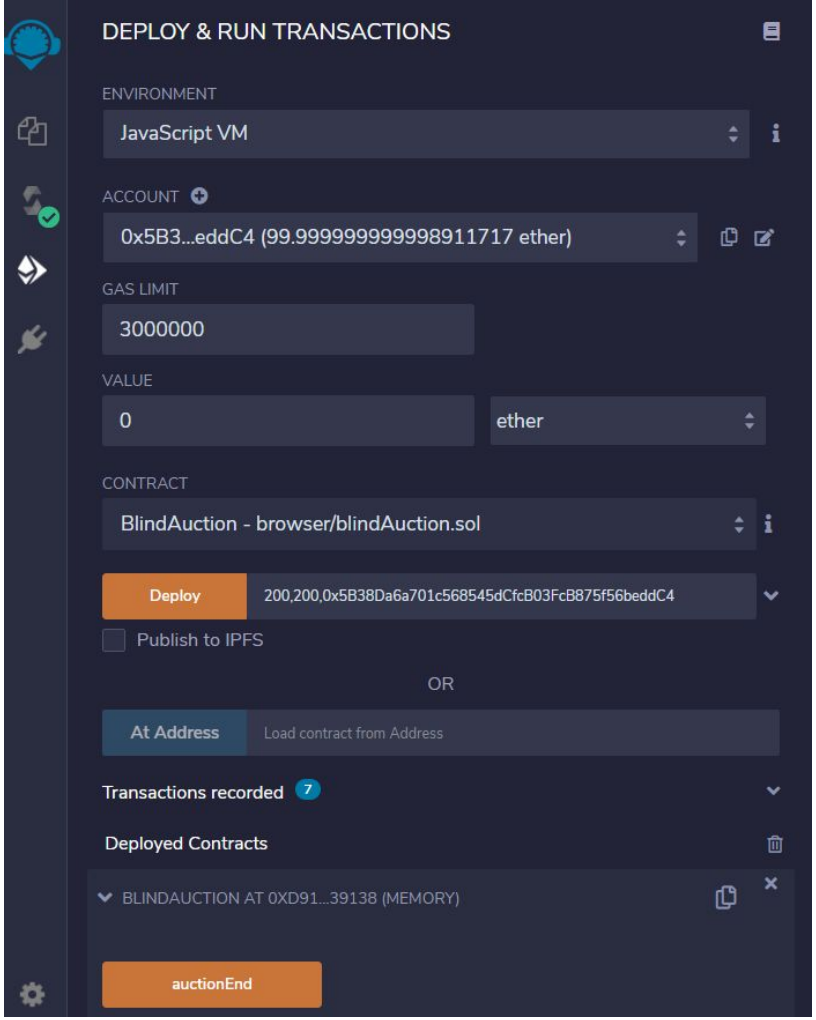
Deployed Contracts

BLINDAUCTION AT 0XD91...39138 (MEMORY)

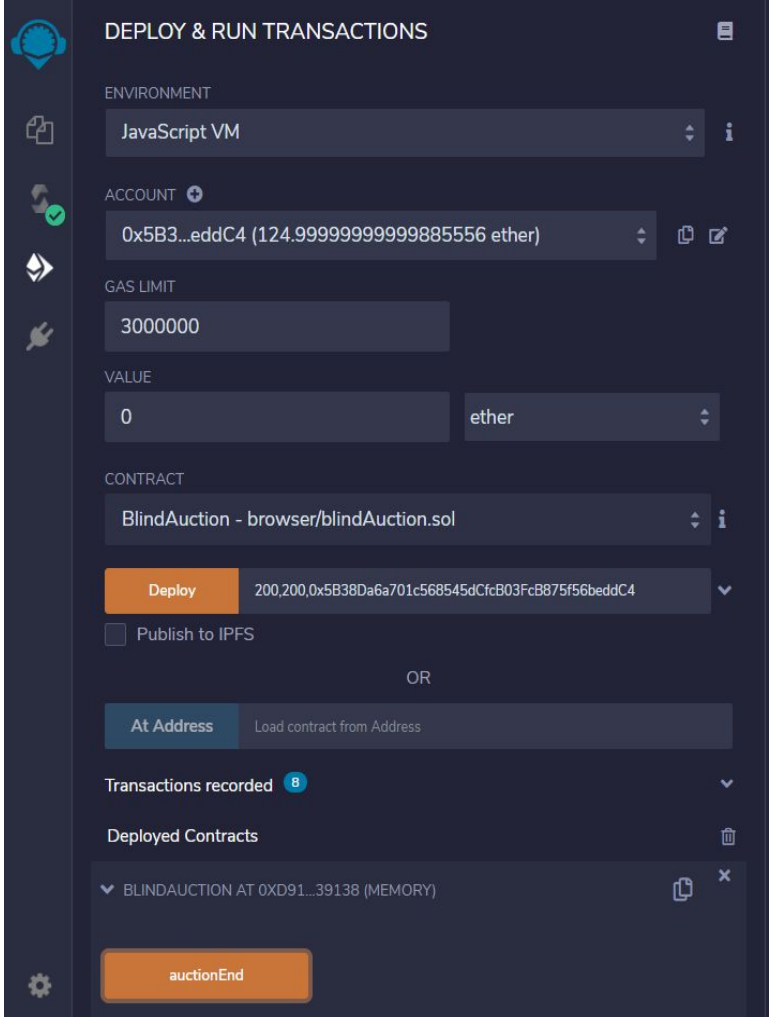
auctionEnd

bid 0x444b7ae584bcd1c8e7fffd44ee520edd520356d684ad2

Placing Blind Bid



Before clicking on auctionEnd button



After clicking on auctionEnd button

[vm] **from:** 0x5B3...eddC4 **to:** BlindAuction.auctionEnd() 0xd91...39138 **value:** 0 wei **data:** 0x2a2...4f46c **logs:** 1 **hash:** 0x8cd...396f6

Debug

status	true Transaction mined and execution succeed
transaction hash	0x8cd581b1206f1d2d2db5df6d1e695bb811bfdb37d45c257d3c7786de6a6396f6
from	0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to	BlindAuction.auctionEnd() 0xd9145CCE52D386f254917e481eB44e9943F39138
gas	3000000 gas
transaction cost	56157 gas
execution cost	34885 gas
hash	0x8cd581b1206f1d2d2db5df6d1e695bb811bfdb37d45c257d3c7786de6a6396f6
input	0x2a2...4f46c
decoded input	{}
decoded output	{}
logs	[{ "from": "0xd9145CCE52D386f254917e481eB44e9943F39138", "topic": "0xdaec4582d5d9595688c8c98545fdd1c696d41c6aeaeb636737e84ed2f5c00eda", "event": "AuctionEnded", "args": { "0": "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db", "1": "2500000000000000000", "winner": "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db", "highestBid": "2500000000000000000" } }]
value	0 wei

Transaction details of AuctionEnd

Conclusion and Future work

- We presented a Blockchain based E-Auction developed on Ethereum.
- We detailed about the security risks of centralised auction mechanisms and how it can be resolved using smart contracts to ensure electronic seals confidentiality, non-repudiation, and immutability.
- We defined smart contracts for both simple and Blind auction and developed a UI to interact with Simple auction smart contract.
- Future work-> Extend this smart contract-based framework to other Blockchains and to improve the security by introducing authentication mechanisms for bidders.

Reference

1. Ilichetty S Chandrashekar, Y Narahari, Charles H Rosa, Devadatta M Kulkarni, Jeffrey D Tew, and Pankaj Dayama. Auction-based mechanisms for electronic procurement. IEEE Transactions on Automation Science and Engineering, 4(3):297–321, 2007
2. S. Underwood, “Blockchain beyond bitcoin,” Commun. ACM, vol. 59, no. 11, pp.15–17, Oct. 2016. [Online]. Available: http://doi.acm.org/ 10.1145/2994581
3. Ethereum smart contract platform. [Online]. Available: https://www.ethereum.org/
4. A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, “Blockchain for iot security and privacy: The case study of a smart home,” in 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), March 2017, pp. 618–623.
5. E.-O. Blass and F. Kerschbaum. Strain: A secure auction for blockchains. In 23rd European Symposium on Research in Computer Security, ESORICS’18, LNCS, 2018.
6. K. Omote and A. Miyaji. A practical English auction with one-time registration. In V. Varadharajan and Y. Mu, editors, ACISP, volume 2119 of LNCS, pages 221–234, 2001..
7. Opensea an Online market place Available: https://opensea.io/
8. Yi-Hui Chen, Shih-Hsin Chen, Iuon-Chang Lin,Blockchain based Smart Contract for Bidding System, IEEE International Conference on Applied System Innovation 2018 IEEE ICASI 2018- Meen, Prior Lam (Eds)