A Practical Training/Minor Project Report

on

# Blockchain based Bidding System

Submitted by

**Bavya Balakrishnan**
**192IT003**
**III Sem M.Tech (IT)**

in partial fulfillment for the award of the degree
of

**MASTER OF TECHNOLOGY**
**in**
**INFORMATION TECHNOLOGY**

**DEPARTMENT OF INFORMATION TECHNOLOGY**
**National Institute of Technology Karnataka, Surathkal**
**July-November 2020**

# CERTIFICATE

This is to certify that the project entitled **Blockchain based Bidding System** is a bonafide work carried out under my guidance for the course IT891/IT897, by **Bavya Balakrishnan,** student of III Sem M.Tech (IT) at the Department of Information Technology, National Institute of Technology Karnataka, Surathkal, during the academic year 2020-21, in partial fulfillment of the requirements for the award of the degree of Master of Technology in Information Technology, at NITK Surathkal.

Place: NITK, Suratkal
Date: 20-11-2020

Signature of the Guide
Name of Guide: Dr. Geetha V

Designation of the Guide: Assistant Professor

Name and Address of Organization: NITK, Suratkal

# DECLARATION

I hereby declare that the project entitled **Blockchain based Bidding System**, submitted by me for the course IT891/IT897 as part of the partial course requirements for the award of the degree of Master of Technology in Information Technology at NITK Surathkal is my original work. I declare that the project has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles elsewhere.

Name and Signature of the Student: Bavya Balakrishnan

Place: NITK Suratkal

Date: 20-11-2020

## Abstract

E- auction is a very popular and widely happening E-commerce activity nowadays through which bidders can bid the goods and commodities directly over the internet. In traditional auction systems third party which can never be guaranteed as a trusted authority plays an important role and act as mediator between seller and buyer. To address this issue, blockchain based auctions were introduced which ensures privacy, security and non-repudiation with the help of smart contracts. In this project we examine how blockchain based Open Auctions (both Simple and Blind Auctions) work in Ethereum.
Index Terms: E-auction, Bid, Beneficiary, Ethereum, Blockchain, Smart Contract

# Contents

# List of Figures

# List of Abbreviations

| | |
|---|---|
| ABI | Application binary interfaces |
| CSS | Cascading Style Sheets |
| dApp | Decentralized Application |
| HTML | Hypertext Markup Language |
| MPC | Multi-Party Computation |
| UI | User Interface |
| VM | Virtual Machine |
| ZKP | Zero-Knowledge Proofs |

# 1 Introduction

An auction is a process where people participating in it make larger and larger bids for the commodity until it gets sold to the person who is ready to pay the most. It involves seller and multiple bidders. There are various Internet auction sites with different formats and rules. Auction theory deals with research on predicting auction formats, optimum bid etc to improve efficiency and profit. E-Auction integrates the internet and ordinary auction to significantly reduce the transaction and transportation cost.

In this project we are implementing a smart contract based open auction. 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. There is a fixed time limit for the auction. Participants are refunded if they do not win the auction. In the end the bid amount of winner is transferred to beneficiary. We are focusing on Simple auction and Blind auction. In blind auction the bidders take the hash of bid amount to send it to auction authorities.

There is another variety of auctions which we call Private auction where the participation is restricted to selected bidders only. It gives an exclusive preference to a group of adversaries before it is made available to the open marketplace.

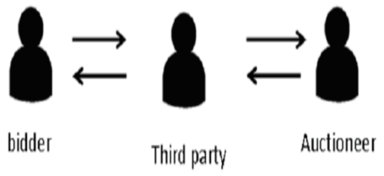


Figure 1.1: Participants in E-auction

The main participants in traditional E-auction are illustrated in Figure 1.1. Auctioneer (or beneficiary) is the person who conducts auction and announce if the good is sold to the winner. Here, the functionalities such as 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. However, this has serious issues like privacy leakage when the personal data and transaction records are stored in a centralised database. In order to resolve this issue, we use Blockchain. 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.

This paper is organized as follows. Brief reviews on Traditional auction system and Blockchain is given in Section 2. How do we integrate the Blockchain to create an efficient E-auction is explained in Section 3. By analysing the experimental results given in Section 4 we draw the Conclusions in Section 5.

# 2 Literature Survey

This section consolidates the relevant and recent works in Open auctions. Various challenges the existing systems and the works to be done to address them are detailed in the Outcome of Literature survey. It is followed by problem statement and objectives for the project.

## 2.1 Background

### 2.1.1 Traditional Bidding System

Traditional bidding system can be mainly classified into 2 namely Public auction or simple auction and Blind action[1]. In public bid the current highest bid is always visible to the Bidders and accordingly they can raise the price to win the product. Thus, the bidding price goes on increasing until nobody is willing to pay a higher price or the time is expired. In Public auction bidders can bid multiple times. A blind auction has some unique features. Instead of sending the actual bid the bidder sends a modified version of their bid amount hashed with a secret key. This involves no competition or pressure towards the end of auction. Once the bidding time is over, the bidders are given a chance to reveal their bids. The valid highest bid will be considered as the winner. The issue faced by Blind auction is that it cannot always ensure if the bid price has been leaked by a third party or an adversary before the reveal time starts.

The protocol presented by Omote and Miyaji [6] talks about an 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. At the end of auction, auction manager declares the winner after verifying its identity by contacting registration manager. This protocol satisfies Anonymity, Traceability, No framing, Unforgeability, Fairness, Verifiability and Efficiency. In this protocol, it is easy to revoke a bidder: RM has only to delete a bidder from RM's bulletin board. The main issue with this protocol is that Registration manager and Auction manager are centralised in nature.

### 2.1.2 Blockchain for E-Auction

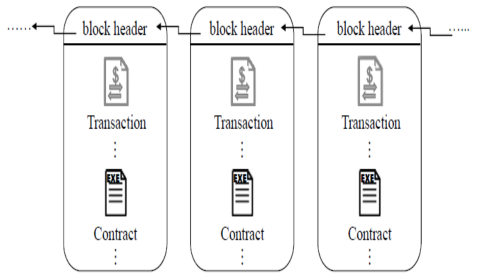


Figure 2.1: Illustration of Blockchain [3]

Blockchain is basically a decentralized, distributed and immutable database ledger that stores many transactions organized by timestamps across peer-to-peer (P2P) network[2]. Blockchain technology can open on to new opportunities and improve the businesses through transparency, enhanced security, and easier traceability. As illustrated in Figure 2.1 Blockchain contains blocks of transactions and smart contracts with each block containing hash of its previous block to form a chain. Every transaction is signed digitally and validated by miners in the network. Smart contract is considered as a special account in Blockchain with associated code and data and it establishes agreement between 2 entities based on predefined rules without depending on any trusted third party.

In [8] authors provided an E-auction mechanism based on blockchain to ensure electronic seals confidentiality, non-repudiation, and unchangeability. With decentralized access structure they proposed, all bidders can bid the product by calling the open contract's trading contract without the involvement of any intermediate brokers. In the paper they couldn't handle the authentication of various bidders as some may call the wrong contract function. For example, the bidder accidentally calls Reveal() to open all bids, due to which the the bidding gets terminated and re-arranged. They figured out the necessity to provide an authority judgment for different functions and determine if the caller can perform the function before actually performing the function.

In [5] authors presented STRAIN (Secure Auctions for Blockchains). Using blockchain and cryptographic primitives like Multi-Party Computation (MPC) and Zero-Knowledge

Proofs (ZKP) they introduced a sealed bid auction system integrated with security proofs to guarantee bid confidentiality against fully-malicious bidders. They introduced a 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.

Opensea [7] provides an efficient blockchain based public auction system. It is a marketplace where products like gaming items, digital art etc can be bought and sold. Bidders are allowd to bid with any amount, with no constraint to be higher than the highest bid. Sellers have the privilege to end the auction at anytime and accepting any bid, not necessarily the highest one. Bidders are also given the freedom to cancel their bids at any time. Unlike English auction, this protocol does not guarantee non-cancellation of bids, neither that the highest bidder wins.

## 2.2 Outcome of Literature survey

Most of the traditional bidding systems are based on Centralised approach which requires high transaction cost. In addition, it cannot guarantee the trust-worthiness of the third-party. It is a single point of failure and easy to compromise. To resolve this issues, the blockchain technology with low transaction cost,enhanced security and privacy is used to develop the smart contract for public bid and blind bid. But even with Blockchain technology researchers are now focused on improving scalability and handling authentication of bidders.

## 2.3 Problem Statement

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

## 2.4 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

## 3 Methodology

In this section we explain how to design Open auction in Ethereum Blockchain. We create smart contract for both Simple auction and Blind auction in Solidity. The smart contract will be acting as an agreement between the buyer and seller during the selling of item remotely in E-Auction. In addition to that we build a UI for Simple auction. We use web based Remix tool for compiling solidity smart contract. A smart contract will be associated with a 20 byte address to identify itself.

### 3.1 Simple Auction

Here we explain the implementation details of Simple auction tool. During the predetermined bidding period everyone can send their bidding amount. Their transactions will include ether to bind the bidders to their bid. 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. In order to make this refund secure, the previously highest bidder must manually call the function to withdraw their money. Those who do not raise any highest bid in the auction will get reverted immediately. We keep a variable which is set to true at the end of Auction to disallow any change in the auction. Once the bidding period is over beneficiary or seller manually call the contract to receive his money.

In Algorithm 1 we provide the the function bid. It is made payable to enable the function to receive ether. We revert the call to bid if the bidding period is over. At any time if we get a highest bid, sending the money back to previous highest bidder by simply using highestBidder.send(highestBid) is not secure and safe because it can be prevented by the caller. It is safer to let the recipient to withdraw their money themselves. The withdraw function(Algorithm 2) is for this purpose. When a bidder calls this function it is important to set the pending returns to zero because the recipient can call withdraw again as part of the receiving call before ‘send‘ returns.

Once we define the smart contract in remix we must invoke it. We instantiate the contract using the Ethereum javascript library using below commands.

**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
```

$var\ SimpleAuction = eth.contract(SimpleAuctionABI, SimpleAuctionBytecode,$
$from : accounts[0], gas : 3000000);$
From the details in Compile tab of remix, we get the ByteCode and ABI. Once we have simple Smart Contract instantiated, we can take inputs from the UI required for invoking the smart contract and then call the required function. The UI is created using HTML. To style the HTML document we use CSS. CSS describes how HTML elements should be displayed.

## 3.2 Blind Auction

The blind auction is very similar to simple auction but with some unique features. Bidder will be sending the hashed version of actual bid. There is no much pressure for bidders to raise higher and higher bid as we move towards the end of auction. Here also we must allow the bidders who didn't win to withdraw their money. In addition to biddingTime we must predefine the revealTime as well. Bidding has to be performed before biddingTime expires. The blinded bid is defined as
'_blindedBid' = keccak256(abi.encodePacked(value, fake, secret)).
A bidder is allowed to place multiple bids. The bid is considered as valid if the ether you send along with the bid transaction is value and fake is set to false. If you need to hide your bid you can send a non exact amount after setting fake to true. In order to ensure security smart contract restricts the refunds to only the bidder who can reveal its blinded bid. The bid function is illustrated in Algorithm 3.

**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 $biddingEnd > currentTime$ or $revealEnd < currentTime$ 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 $bidToCheck.deposit \geq value$ then
    if $value > highestBid$ then
      refund -= value, pendingReturns[highestBidder] += highestBid
      highestBid = value, highestBidder = bidder
    end if
  end if
  bidToCheck.blindedBid = bytes32(0);
end for
$msg.sender.transfer(refund)$
```

Once the bidding ends the next next phase starts. Reveal function(Algorithm 4) can be called till the revealTime is over. Refunds will be available for all topped bids and invalid bids that were blinded properly. If the bidder raised a bid which is highest so far it is not refunded for the time-being. Similar to the simple auction, we use withdraw function for withdrawing a topped bid. Once the revealEnd is reached then the can manually end the auction and the winning bid amount will be send to the beneficiary.

## 3.3 Planned Enhancements

- Test the framework on Ganache,personal blockchain for Ethereum development.
- Test the framework by creating private blockchain using multiple devices
- Develop a UI based dApp for Simple auction

# 4 Implementation

## 4.1 Work done

Smart contracts for both simple auction and blind auction are developed using Remix IDE for Solidity. 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.

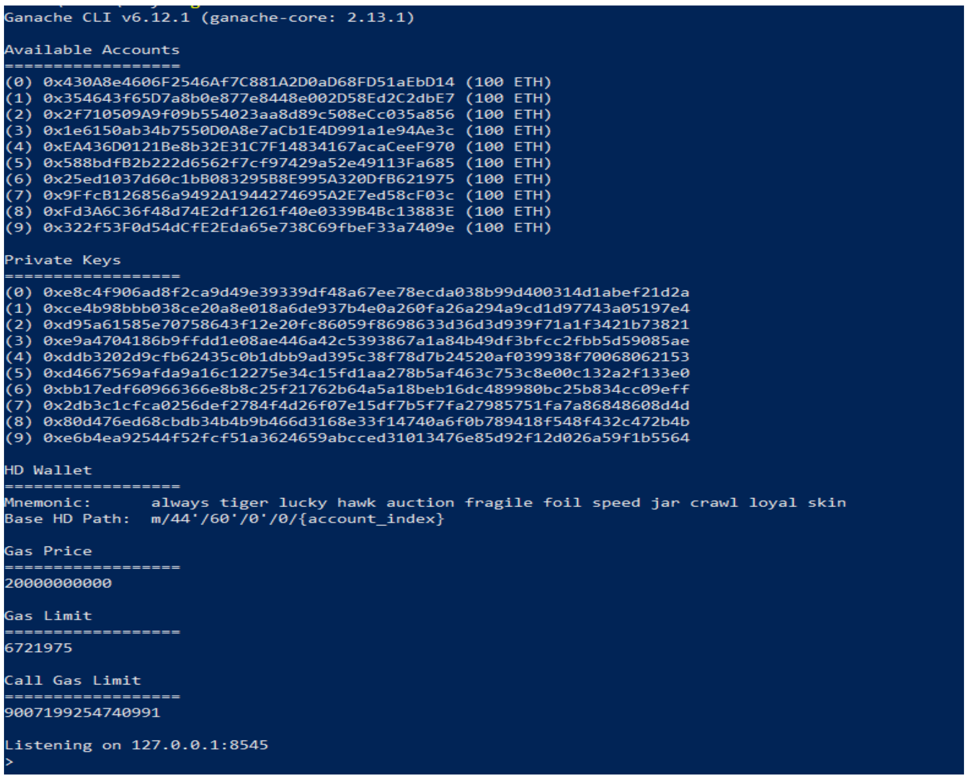
To test the entire framework we use ganache-cli which is a personal blockchain for Ethereum development. It can be used to deploy contracts, develop and evaluate applications. We included ethereumjs-testrpc in the html script to simulate full client behaviour. Along with that we used ethjs which is primarily designed for building light-weight dApps to act as simple javascript interface for Ethereum nodes and clients. We can install the ganache cli using npm using the command: $npm\ install - g\ ganache - cli$
When you start the ganache ethereum test blockchain it will be running in localhost 8545 port. Once we setup a local Ethereum node using the ganache cli, we need to connect it to the Ethereum block chain. To do that we select the Run tab in Remix and select Web 3 provider in the Environment. This prompt you for connecting to your local ethereum node and you need to enter the address $http : //localhost : 8545$ where the Ethereum node is running. Now we can see changes in the ganache-cli when the local host becomes a part of the global Ethereum blockchain.

For blind auction we developed the smart contract and its deployed and run on JavaScript VM. This VM provides 15 ethereum accounts with 100 ether balance in each. We provide the biddingTime,revealTime,beneficiary address during deployment. Every bidder can provide their blindedbid through bid function to place the bid after specifying the value in ether. Once the bidding ends bidders can start reveal the bids and accordingly smart contracts decide the highest bidder and declare it as winner.

I also tested both the smart contracts by creating a private blockchain using 2 laptops. After installing geth on both the devices I initialized the ethereum nodes using the configuration details mentioned in genesis.json. With the help of admin.addPeer() commands we can create a peer to peer network. The connections between nodes are valid only if peers have identical protocol version and network ID. By giving a non default network id we can effectively isolate our network. Thus, whenever new blocks are mined and added in Node 1, this new block could propagate over the private blockchain network, and the local blockchain copy of node 2 will be updated automatically.

## 4.2 Results and Discussion

### 4.2.1 Simple Auction



```
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
>
```

Figure 4.1: Ganache cli listening on port 8545

For creating Simple auction smart contract we used Ganache cli. It uses ethereumjs to simulate the behavior of full client and to make faster, easier and safer Ethereum applications . When we start Ganache cli we see the localhost listening on port 8545 as shown in Figure 4.1. Then using web3 provider we connect out local node to ethereum global blockchain. Once we instantiate the smart contract using the javascript we see the deployed contract as shown in Figure 4.2. Figure 4.3 shows the UI of the DAPP. 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. Bidders can place their bid. Exceptions are thrown when any of the following happens.

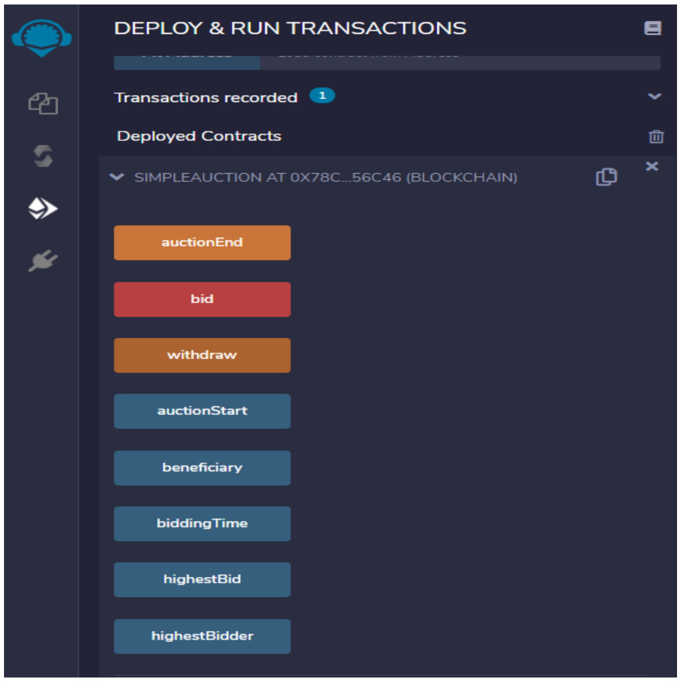
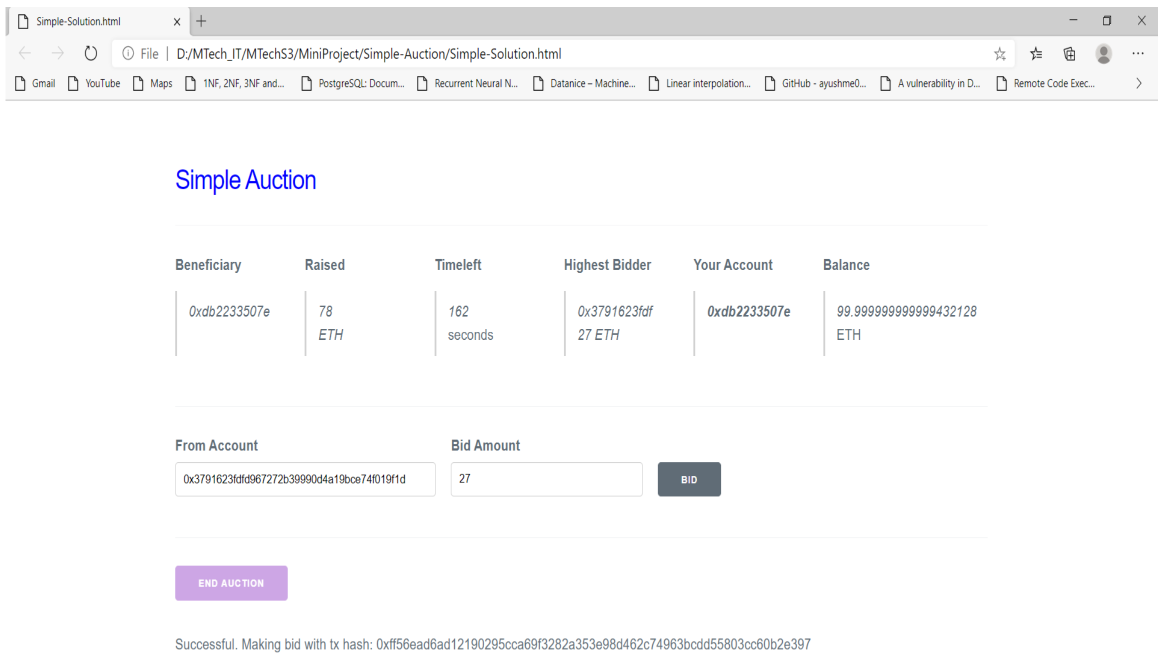


Figure 4.2: Simple Auction smart contract

- 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

*Raised* field in the UI shows the value in ether owned by smart contract. This will be the sum of topped bids. Once the bidding period is over and ENDAUCTION button is pressed then the highest bid is sent to the beneficiary and the smart contract will be left with the bids of all topped bids. They must be withdrawn by bidders themselves



Simple Auction

| Beneficiary | Raised | Timeleft | Highest Bidder | Your Account | Balance |
|---|---|---|---|---|---|
| *0xdb2233507e* | *78 ETH* | *162* seconds | *0x3791623fdf 27 ETH* | ***0xdb2233507e*** | *99.999999999999432128* ETH |

From Account: 0x3791623fdfd967272b39990d4a19bce74f019f1d

Bid Amount: 27

Successful. Making bid with tx hash: 0xff56ead6ad12190295cca69f3282a353e98d462c74963bcdd55803cc60b2e397

Figure 4.3: UI of the Decentralized Simple Auction App

### 4.2.2 Blind Auction

The structure of Blind auction smart contract is given in Figure 4.4. It is deployed using Javascript VM by providing input values of bidding time,reveal time and address of beneficiary. 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

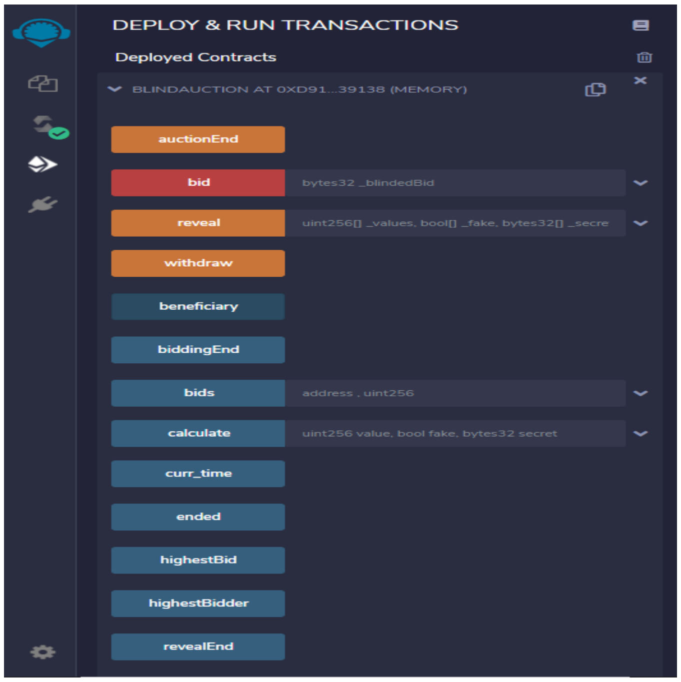
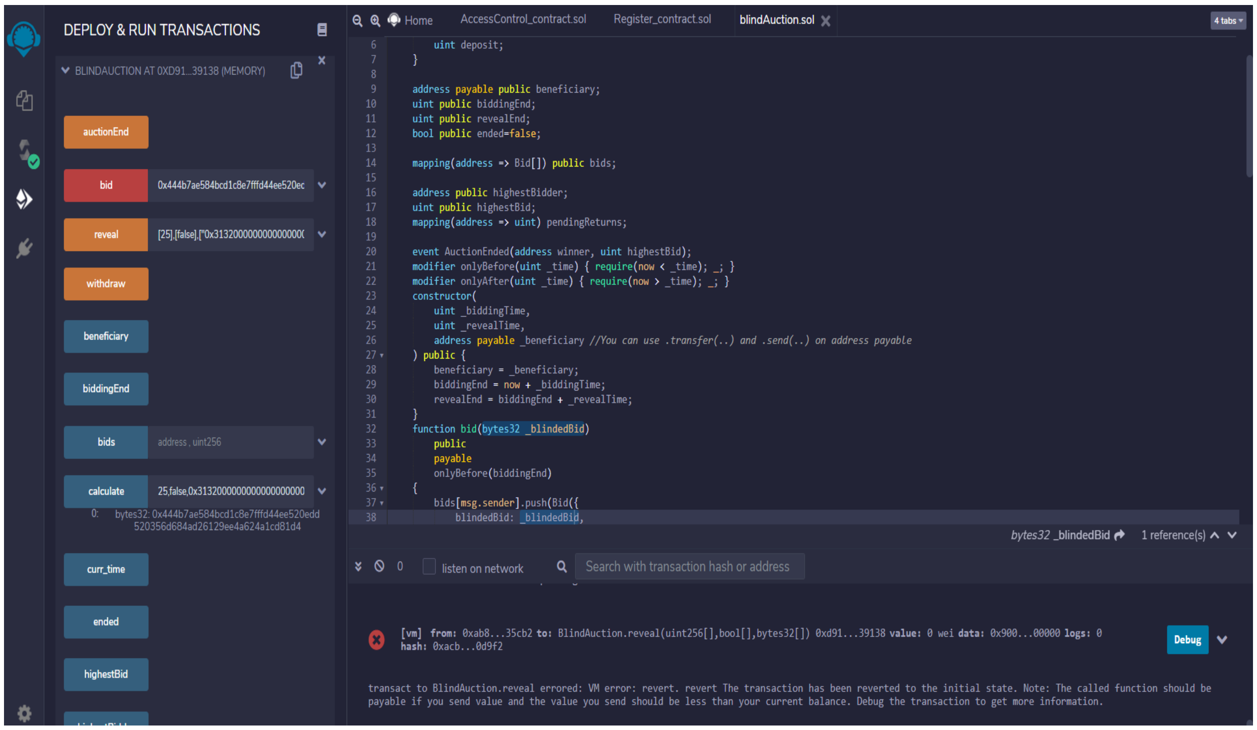


Figure 4.4: Blind Auction smart contract

Figure 4.5 illustrates the reverted transaction while we try to reveal the blinded bids before the bidding period is over. It throws the error *"transact to BlindAuction.reveal errored: VM error: revert. The transaction has been reverted to the initial state"*

I deployed the Blind auction on a private blockchain as well. Using the web3 provider the remix IDE on web browser is connected to the Private blockchain network. The smart contract deployed on Private network 2113 is shown in Figure 4.6. We shouldn't forget to start the mining while performing transactions so that it gets included in Blockchain. Figure 4.7 shows the details of transaction sent to end the blind auction after the reveal period is over.

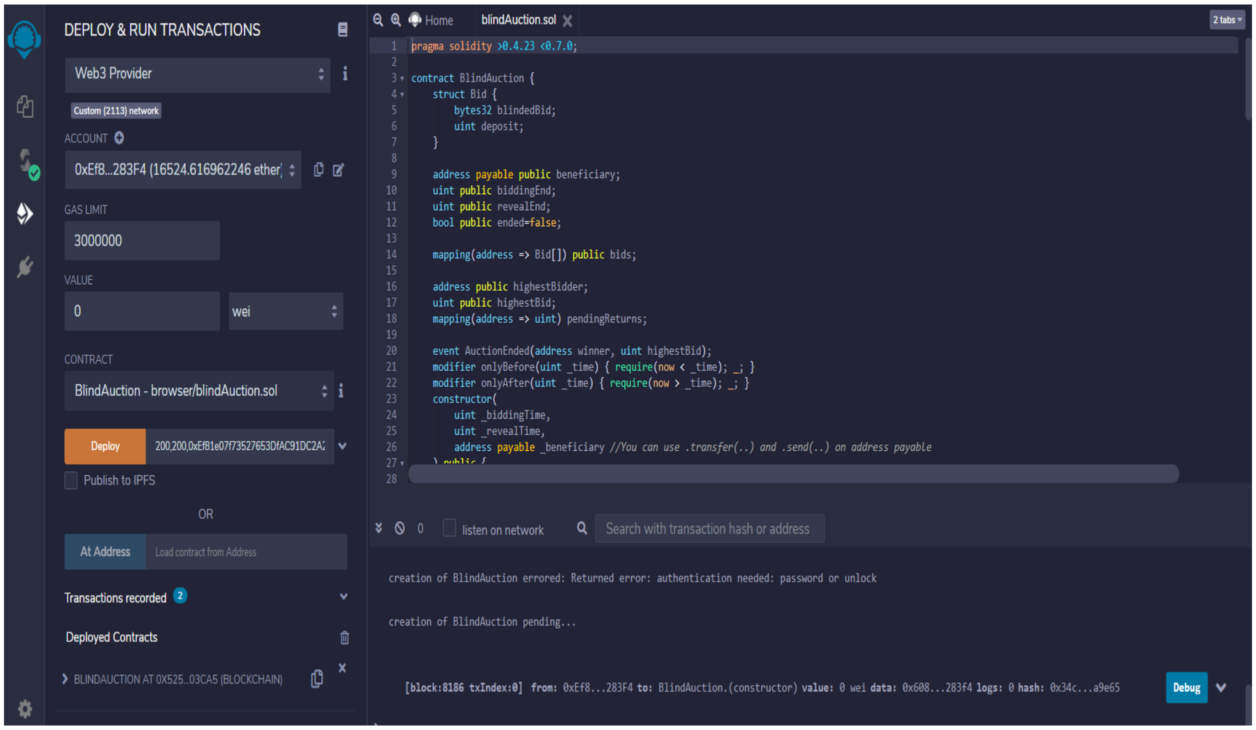


```
bid        0x444b7ae584bcd1c8e7fffd44ee520ec
reveal     [25],[false],["0x31320000000000000000(
bids       address , uint256
calculate  25,false,0x31320000000000000000000
   0:  bytes32: 0x444b7ae584bcd1c8e7fffd44ee520edd
               520356d684ad26129ee4a624a1cd81d4
```

```
        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 {
        beneficiary = _beneficiary;
        biddingEnd = now + _biddingTime;
        revealEnd = biddingEnd + _revealTime;
    }
    function bid(bytes32 _blindedBid)
        public
        payable
        onlyBefore(biddingEnd)
    {
        bids[msg.sender].push(Bid({
            blindedBid: _blindedBid,
```

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

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.
```

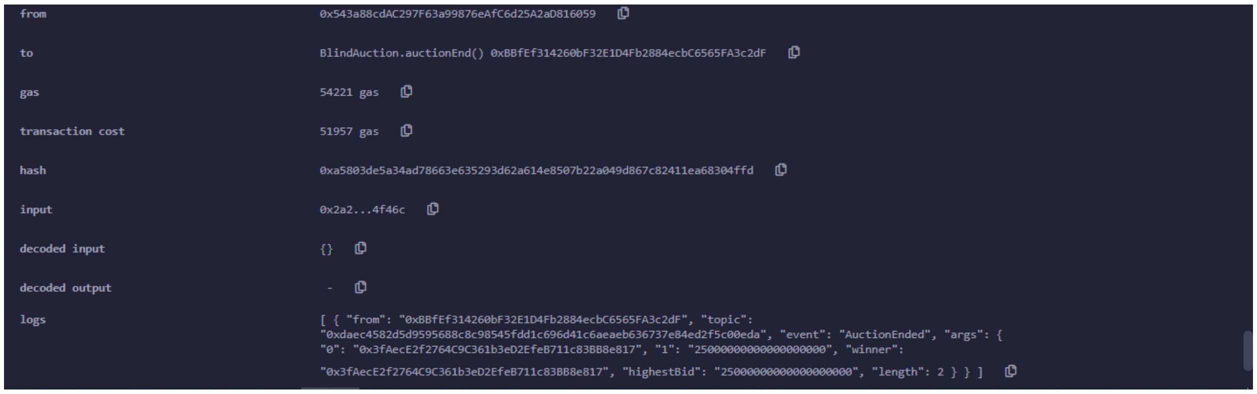
Figure 4.5: Reverted Transaction when reveal is called before bidding ends


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
listen on network
Search with transaction hash or address
Debug

```
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 {
```

```
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
```

Figure 4.6: Blind auction deployed on Private Blockchain



```
from                  0x543a88cdAC297F63a99876eAfC6d25A2aD816059
to                    BlindAuction.auctionEnd() 0xBBfEf314260bF32E1D4Fb2884ecbC6565FA3c2dF
gas                   54221 gas
transaction cost      51957 gas
hash                  0xa5803de5a34ad78663e635293d62a614e8507b22a049d867c82411ea68304ffd
input                 0x2a2...4f46c
decoded input         {}
decoded output         -
logs                  [ { "from": "0xBBfEf314260bF32E1D4Fb2884ecbC6565FA3c2dF", "topic":
                      "0xdaec4582d5d9595688c8c98545fdd1c696d41c6aeaeb636737e84ed2f5c00eda", "event": "AuctionEnded", "args": {
                      "0": "0x3fAecE2f2764C9C361b3eD2EfeB711c83BB8e817", "1": "25000000000000000000", "winner":
                      "0x3fAecE2f2764C9C361b3eD2EfeB711c83BB8e817", "highestBid": "25000000000000000000", "length": 2 } } ]
```

Figure 4.7: Details of AuctionEnd transaction in Blind auction

## 5 Conclusion and Future work

In this project 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.

As a future work I would like to extend this smart contract-based framework to other Blockchains and to improve the security by introducing authentication mechanisms for bidders.

## References

[1] llichetty 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)