Designing Smart-Contract Based Auctions

Chapter · January 2020

DOI: 10.1007/978-3-030-16946-6_5

4 authors, including:

Stelvio Cimato
University of Milan
131 PUBLICATIONS **1,368** CITATIONS
SEE PROFILE

Ernesto Damiani
Khalifa University
810 PUBLICATIONS **11,255** CITATIONS
SEE PROFILE

Designing smart-contract based auctions

Chiara Braghin[1], Stelvio Cimato[1], Ernesto Damiani[1,2], and Michael Baronchelli[1]

[1] Dipartimento di Informatica, Università degli Studi di Milano, Milano, Italy
{chiara.braghin,stelvio.cimato,ernesto.damiani}@unimi.it
michael.baronchelli@studenti.unimi.it
[2] Centre on Cyber-Physical Systems, Khalifa University, Abu Dhabi, UAE
ernesto.damiani@kustar.ac.ae

Abstract. In this paper, we developed an online auction system based on Ethereum smart contracts. A smart contract is executable code that runs on top of the blockchain to facilitate, execute and enforce an agreement between untrusted parties without the involvement of a trusted third party. A decentralised auction guarantees greater transparency and avoids cheating auctioneers. Since in Ethereum computation is expensive as transactions are executed and verified by all the nodes on Ethereum network, we analysed our implementation in terms of cost and time efficiency, obtaining promising results.

Keywords: Auction · Blockchain · Smart contract.

1 Introduction

Auctions are platforms for selling goods in a public forum through open and competitive bidding. Commonly, the auction winner is the bidder who submitted the highest price, however, there are a variety of other rules to determine the winner. The auctioneer leads the auction according to the rules and always charge a fee to the vendor for his services, usually a percentage of the gross selling price of the good.

In electronic auctions (*e-auctions*), the interaction among auctioneers, suppliers and buyers takes place by means of a centralised intermediary providing a platform for posting products, checking bids validity, and committing the winner. Such a third party, that could be impersonated by the auctioneer himself, may ask a commission as well, and has to be trusted.

In this paper, we design and implement the most common types of auctions by means of Ethereum smart contracts. Ethereum is the second most popular blockchain, with a built-in Turing complete programming language that allows running smart contracts in a global virtual machine environment known as Ethereum Virtual Machine (EVM), without depending on any third-party. The rules required by the different types of auctions and the actions of the different actors can be coded directly in a smart contract. Thus, by inheriting some properties from the blockchain, in our proposed solution:

1. every transaction executed in the smart contract is visible and verifiable to the entire network: bidders and the auctioneer cannot cheat and the winning bidder can use the blockchain as a proof;
2. a trusted third party is not required, promising low transactions fees compared to traditional systems;
3. neither the code, nor the data stored in the blockchain can be modified;
4. the identity of the actors is always verified before any action is taken.

However, in Ethereum, computation is expensive as transactions are executed and verified by all the nodes on Ethereum network. Therefore, Ethereum defines a *gas* metric to measure the computation efforts and storage cost associated with transactions. That is, each transaction has a fee (i.e., consumed gas) that is paid by the transaction's sender in *Ether* (Ethereum currency). There is also a block gas limit that defines the maximum amount of gas that can be consumed by all transactions combined in a single block. Therefore, smart contracts cannot include very expensive computations that exceed the block gas limit.

In this paper, our main goal was to evaluate the cost of a prototype implementation of the four classical types of auction (English auction, Dutch auction, First-price sealed-bid auction, and Second-price sealed-bid auction). Thanks to the promising experimental results, we plan to formally verify the correctness of the smart contracts, and to further investigate the security and privacy issues on the blockchain.

The structure of this paper is as follows. Section 2 discusses background information about auctions, blockchain and smart contracts technologies. Section 3 describes how smart contracts for auctions were designed, developed, and tested, and discusses the experimental results we obtained, in terms of gas costs and time. Section 4 concludes the paper.

2 Background and Related Work

This section presents some background information on auctions, blockchain and smart contracts technologies. It also describes the few current works presenting auction systems based on blockchain technologies.

2.1 Auction

Auctions can be described as games of incomplete information that involve a certain number of actors: a *vendor* V, willing to sell a good, an *auctioneer* A, leading the auction according to some predefined rules and asking a commission for his services, and a set of *bidders* B, willing to buy the good at the best price.

There exist traditionally four types of auctions [5] for the allocation of a single item, differing on the rules to determine the winner, or the final buying price:

– *English auction*, also called *open ascending price auction*, where participants bid openly against one another, with each subsequent bid required to be

higher than the previous bid. There are many variants depending on how bidders signal their will to attend and to go on with attending, or on the way the subsequents bids are given. For example, an auctioneer may announce prices raising them with small increments as long as there are at least two interested bidders, or bidders may call out their bids themselves (or have a proxy call out a bid on their behalf). The auction ends when a single bidder remains signaling his interest, at which point the highest bidder pays his bid. The seller may also define a *reserve price*, that is the minimum price he will accept as the winning bid in the auction.
- *Dutch auction*, also called *open descending price auction*, used for perishable commodities such as flowers, fish and tobacco. In the traditional Dutch auction, the auctioneer begins with a high price and then continously lowers it until a bidder is willing to accept the auctioneer's price, or until the seller's reserve price is met.
- *First-price sealed-bid auction* (FPSB), also called *blind auction*, where all bidders submit bids in a sealed envelop to the auctioneer, so that no bidder knows the bid of any other participant. Later, the auctioneer opens the envelope to determine the winner who submitted the higher bid.
- *Vickrey auction*, also known as *Second-price sealed-bid auction* (SPSB), which is identical to the first-price sealed-bid auction, with the exception that the winning bidder pays the second-highest bid rather than his own.

2.2 Blockchain and smart contract

A blockchain [6] is a distributed peer-to-peer network where non-trusting members can interact with each other without a trusted intermediary, in a verifiable manner. Smart contracts are scripts that reside on the blockchain that allow for the automation of multi-step processes. In this section, we give the details of how they work in order to be able to understand the implementation details that will be given in the next section.

Blockchain technology. A blockchain is a *distributed ledger*, typically managed by a peer-to-peer network collectively adhering to a protocol for inter-node communication and for validating new blocks. It is distributed in the sense that there is no central repository of information, no database on a file server which can be hacked: it is a distributed data structure that is replicated and shared among all the members of a network. It is a ledger in the sense that it records and stores all transactions that have ever occurred in the blockchain network in a permanent and verifiable way.

In a blockchain, each block is identified by its cryptographic hash. Each block contains the cryptographic hash of the previous block, resulting in a chain of blocks (see Figure1), a timestamp, and a set of transactions (generally represented also as a Merkle tree root hash in the block header). Thus, by design, a blockchain is resistant to modification of the transaction data since, once recorded, the data in any given block cannot be altered retroactively without alteration of all subsequent blocks.

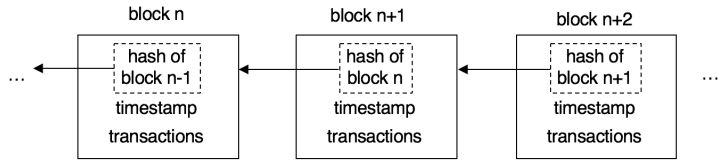

Fig. 1. Structure of a blockchain.

A blockchain network consists of a set of nodes that operate on the same blockchain via the copy each one holds. Users interact with the blockchain via a pair of private/public keys: they use their private key to sign their own transactions, and they are addressable on the network via their public key.

Transactions that occurred in the network are verified by special nodes (called *miners*). Verifying a transaction means checking the sender and the content of the transaction. Miners generate a new block of transactions after solving a computationally expensive task (called *Proof of Work*) and then propagate that block to the network. Other nodes in the network can validate the correctness of the generated block.

Regardless of specific implementations, a blockchain gives us the following benefits out of the box:

- a blockchain is resistant to modification of data;
- a blockchain is tolerant of node failures;
- a blockchain is a distributed peer-to-peer system able to reach consensus without the need of a third trusted party;
- all the information in the blockchain is available to all the nodes of the network;
- a blockchain is a method for tagging different pieces of information as belonging to different participants, and enforcing this form of data ownership without a central authority.

Smart Contracts The term *smart contract* was introduced by Nick Szabo in 1994 as "a computerized transaction protocol that executes the terms of a contract" [7]. The idea was to translate contractual clauses (collateral, bonding, etc.) into code, and embed them into property (hardware, or software) that could self-enforce them, in order to minimize the need for trusted intermediaries between transacting parties, and the occurence of malicious or accidental exceptions.

Within the blockchain context [8], a smart contract is executable code stored and running on the blockchain to facilitate, execute and enforce the terms of an agreement. We trigger a smart contract by addressing a transaction to it. It then executes independently and automatically in a prescribed manner on every node in the network, according to the data that was included in the triggering transaction (this implies that every node in a smart contract enabled blockchain is running a virtual machine (VM), and that the blockchain network acts as a distributed VM).

Thus, the main features of a smart contract are:

1. a smart contract is deterministic: the same input will always produce the same output;
2. a smart contract is stored on the blockchain, thus it can be inspected by every network participant;
3. since all the interactions with a contract occur via signed messages on the blockchain, all network participants get a cryptographically verifiable trace of the contract's operations.

2.3 Blockchain-based bidding systems

As far as we know, this is the first attempt to implement the four most common types of auctions, and to analyse the implementations in terms of cost and time efficiency. Previous works focused on blind auctions and on security issues, since anonymity and confidentiality seem to collide against the idea of distributed ledger and the inherent transparency and lack of privacy of blockchains.

In [4], the authors implement on top of the Ethereum blockchain a protocol to achieve bids privacy in sealed-bid auctions. In [1], a protocol guaranteeing bid confidentiality against fully-malicious parties is described.

In [3], the authors discuss with little details a possible implementation with Ethereum smart contracts of sealed-bid auctions.

3 Design and implementation of a smart-contract based bidding system

We implemented the four classical types of auction (English auction, Dutch auction, First-price sealed-bid auction, and Second-price sealed-bid auction) by means of four Ethereum smart contracts. In the rest of this Section, we will describe how Ethereum works, then we will show some code fragments implementing the major operations for two of the auctions. Finally, we will evaluate the implementations in terms of cost and time efficiency.

3.1 Ethereum

Smart contracts can be developed and deployed in different blockchain platforms (e.g., Ethereum, Bitcoin and NXT). At the moment, Ethereum [2] is the most popular public blockchain platform for developing smart contracts, since it provides a built-in Turing-complete language called Solidity, that can be used to write any smart contract and decentralised application. The code written in Solidity can be compiled into Ethereum Virtual Machine (EVM) bytecodes to be uploaded (and possibly executed) on the blockchain. Solidity is a contract-oriented, high-level language influenced by C++, Python and JavaScript. It is statically typed, and it supports inheritance, libraries and complex user-defined types among other features.

Ethereum Accounts. Accounts (also called *state objects* or *entities*) play a central role in Ethereum: the state of the Ethereum network consists of the state of all its accounts. There are two types of accounts: *externally owned accounts* (EOAs), and *contract accounts*, which represent users interacting via transactions with the blockchain, or smart contracts, respectively. An external account has an Ether balance, can send transactions (either to transfer Ether, or to trigger contract code), is controlled by private keys, has no associated code. A smart contract has an account balance, a private storage and an associated executable code. Once it is deployed into the blockchain, the contract code cannot be changed. Every time a contract account receives a transaction fired by a user, or a message (a sort of function call) sent by another contract, its code is executed by the EVM on each node. The contract may, based on the transaction it receives, read/write to its private storage, store money into its account balance, send/receive money from users or other contracts, or send messages to other contracts, triggering their execution. The contract's state will then be updated accordingly.

Ether and gas. *Ether* (ETH) is the name of the currency used within Ethereum. It is used to pay for computation within the blockchain and the EVM, although transactions' execution fee is computed in terms of *gas*: one unit of gas corresponds to the execution of one atomic instruction, i.e., a computational step. Gas and ether are decoupled deliberately since units of gas align with computation units having a natural cost, while the price of ether generally fluctuates as a result of market forces.

Transaction and messages. The term *transaction* is used in Ethereum to refer to the signed data package that stores a message to be sent from an externally owned account to another account on the blockchain. Contracts have also the ability to send *messages* to other contracts: messages are virtual objects that exist only in the Ethereum execution environment, they can be conceived of as *function calls*. Essentially, a message is like a transaction, except it is produced by a contract and not by an external actor.

They both contain the recipient, a value field indicating the amount of wei (1 *ether* is 1e18 *wei*) to transfer from the sender to the recipient, an optional data field that represents the input data to the contract, a gasLimit field representing the maximum number of computational steps the transaction or code execution is allowed to take to be used to compute the cost of the computation. Miners can refuse to process a transaction with a gas price lower than their minimum limit. A transaction contains also a signature identifying the user and a gasPrice field, representing the fee the sender is willing to pay for gas.

When a contract is executed as a result of being triggered by a message or transaction, every instruction is executed on every node of the network. This has a cost: for every executed operation there is a specified cost, expressed in a number of gas units. The maximum total ether cost of a transaction is then equal to gasLimit $*$ gasPrice.

3.2 Tools and environment

For the implementation and deployment of smart contracts, we used Remix to write Solidity smart contracts; Ropsten, one of Ethereum public testing networks, to deploy and run the smart contracts; MetaMask to manage the accounts of the bidders, and Etherscan, an online service allowing to explore and search the Ethereum blockchain and its public testnets, to verify the correctness of execution of the smart contracts.

3.3 Implementation details

We have implemented a prototype consisting of four different smart contracts, one for each type of auction. In the rest of the Section, we describe in detail some of the functions in the smart contracts of the English auction and of the First-price sealed-bid auction, since the others only differ slightly.

English auction. In an English auction, participants bid openly against one another, with each subsequent bid required to be higher than the previous bid, starting from a *reserve price* chosen by the seller. At the end, the highest bid wins.

The key aspects that need to be set up by the constructor are (see Figure 2 for a Solidity fragment): (*i*) the beneficiary, an external account representing the seller of the good that starts the auction by creating the contract; (*ii*) endTime, the time after which no further bids are accepted; (*iii*) deliveryLimit, the time lapse by which the winner has to receive the good; (*iv*) reservePrice, the minimum price the seller will accept as bid in the auction; and (*v*) minIncrement, the minimum difference between two bids.

The bidding function bid() is reported in Figure 3. The first four lines of code express the main rules of the auction: a bid must be done before the auctions ends, it must be higher than the previous bid and of the reserve price, and it must come from a valid bidder and not from the seller. If the bid satisfies all the constraints, it will be accepted and recorded as the highest bid.

First-price sealed-bid auction. In a First-price sealed-bid auction (FPSB), all bidders submit bids in a sealed envelop to the auctioneer, so that no bidder knows the bid of any other participant. Later, the auctioneer opens the envelope to determine the winner who submitted the higher bid. In the constructor, the difference with the English auction (see Figure 4) is given by the revealEnd, i.e., the time after the end of the auction when it is possible to reveal the bid.

In this auction, the bid must be sealed. A bidder (see Figure 5) sends his own blinded bid, which is the hash of the concatenation of the actual value of the bid and a secret, used as a salt. In this case the value of the bid cannot be read also by the auctioneer/smart contract, since at this point he does not know the secret used to compute the hash.

```
constructor(
    uint _biddingTime,
    uint _deliveryTime,
    uint _reservePrice,
    uint _minIncrement
) public {
    beneficiary = msg.sender;
    endTime = now + _biddingTime;
    deliveryLimit = endTime + _deliveryTime;
    reservePrice = _reservePrice;
    minIncrement = _minIncrement;
}
```

Fig. 2. English auction: the constructor.

```
function bid() public payable {
    require(now <= endTime);
    require(msg.value >= highestBid+minIncrement);
    require(msg.value >= reservePrice);
    require(msg.sender != beneficiary);

    if (highestBid != 0) {
        highestBidder.transfer(highestBid);
    }
    highestBidder = msg.sender;
    highestBid = msg.value;
    emit NewHighestBid(msg.sender, msg.value);
}
```

Fig. 3. English auction: Bidding function.

```
 constructor(
    uint _biddingTime,
    uint _revealTime,
    uint _deliveryTime,
    uint _reservePrice
) public {
    beneficiary = msg.sender;
    biddingEnd = now + _biddingTime;
    revealEnd = biddingEnd + _revealTime;
    deliveryLimit = revealEnd + _deliveryTime;
    reservePrice = _reservePrice;
}
```

Fig. 4. First-price sealed-bid auction: the constructor.

```
function bid(bytes32 _blindedBid)
    public
    payable
    onlyBefore(biddingEnd)
{
    require(msg.sender != beneficiary);
    bids[msg.sender].push(Bid({
        blindedBid: _blindedBid,
        deposit: msg.value
    }));
}
```

Fig. 5. First-price sealed-bid auction: Bidding function.

3.4 Evaluation of the system

Cost Analysis. As described in Section 3.1, the computation within the EVM has a cost. In this Section, we show the *gas* costs and the corresponding prices in Euro for the deployment and the major operations of the smart contracts implementing the four different types of auctions. At the time of carrying out the experiments, October 2018, the *ether* exchange rate was 1 ETH = 192.80 €, and the median gasPrice was approximately 0.0000002 ETH (20 Gwei).

Table 1. Costs of an *English auction*.

Operation	Gas used	Cost (ETH)	Cost (Euro)
Contract deployment	2345	0.000469	0.09
Bidding	320	0.000064	0.01
End the auction	270	0.000054	0.01
Payment	275	0.000055	0.01

Table 2. Costs of an *Dutch auction*.

Operation	Gas used	Cost (ETH)	Cost (Euro)
Contract deployment	2535	0.000507	0.10
Price decreasing	140	0.000028	0.01
Bidding	410	0.000084	0.02
End the auction	195	0.000039	0.01
Payment	200	0.00004	0.01

Table 3. Costs of an *First-Price Sealed-Bid auction.*

Operation	Gas used	Cost (ETH)	Cost (Euro)
Contract deployment	3635	0.000727	0.14
Bidding	425	0.000085	0.02
Bid opening	370	0.000074	0.01
End the auction	270	0.000054	0.01
Payment	275	0.000055	0.01

Table 4. Costs of an *Second-Price Sealed-Bid auction.*

Operation	Gas used	Cost (ETH)	Cost (Euro)
Contract deployment	3805	0.000761	0.15
Bidding	425	0.000085	0.02
Bid opening	470	0.000094	0.02
End the auction	270	0.000054	0.01
Payment	275	0.000055	0.01

As expected, the smart contract deployment cost is more expensive in the two blind actions, since it involves a larger number of data and of set-up operations. As for the other operations, the cost depends on the complexity of the operation: for example, a sum costs 1 *gas*, whereas the execution of a SHA3 hash costs 20 *gas*.

Time analysis. In Ethereum, a new block is validated every 15 seconds and the transaction rate is of around 20 seconds. During the test in the Ropsten network, the effective average transaction time (i.e., transaction processing, validation and inclusion in a new block) was of 10 seconds. The effective time is the result of the propagation time within all the nodes in the network, and of the time lapse between the creation of two new block. This value may be a problem when bidding just before the end of the auction.

Privacy considerations. At the moment, in our prototype implementation, users do not provide any personal data, since their identity is given by their account address, which is public since Ethereum is a public blockchain. For this reason, our implementation can be considered pseudo-anonymous: a user will be linked to a public Ethereum address, but no one will get to know the actual name or address. However, by observing the blockchain evolution, it is possible to collect all the past transactions by a single user, that can be seen as lack transactional privacy. However, this is not a main issue for some of the auctions we implemented.

4 Conclusion

In this paper, we showed how it is possible to build different types of auctions on top of Ethereum blockchain. We analysed the implementation in terms of

cost and time efficiency. The advantages of a smart-contract based auction with respect to a classical eAuction are:

- *transparency*: the bidding history is public and verifiable by all the users;
- *non-repudiation*: a bidder cannot negate his bid;
- *integrity*: a bidder cannot modify his own bid, or other bidders' bid;
- *no trusted third party.*

For future works, we intend to apply formal verification techniques on our code to detect potential flaws and vulnerabilities. Moreover, we want to further investigate the security and privacy issues.

References

1. Blass, E., Kerschbaum, F.: Strain: A secure auction for blockchains. In: Computer Security - 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part I. pp. 87–110 (2018)
2. Buterin, V.: Ethereum White Paper: A Next-Generation Smart Contract and Decentralized Application Platform (2013)
3. Chen, Y., Chen, S., Lin, I.: Blockchain based smart contract for bidding system. In: 2018 IEEE International Conference on Applied System Invention (ICASI). pp. 208–211 (Apr 2018). https://doi.org/10.1109/ICASI.2018.8394569
4. Galal, H.S., Youssef, A.M.: Succinctly Verifiable Sealed-Bid Auction Smart Contract. In: Garcia-Alfaro, J., Herrera-Joancomart, J., Livraga, G., Rios, R. (eds.) Data Privacy Management, Cryptocurrencies and Blockchain Technology. pp. 3–19. Lecture Notes in Computer Science, Springer International Publishing (2018)
5. Krishna, V.: Auction Theory. Academic Press, 2 edn. (2010)
6. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2008)
7. Szabo, N.: Smart Contracts (1994)
8. Wood, D.G.: Ethereum: a Secure Decentralised Generalised Transaction Ledger (2014)