Two-Stage Market-Based Task Allocation for Blockchain-Based Cyber-Physical Production Systems

Larissa Krämer [0000-0003-1620-3083], Rico Ahlbäumer [0000-0001-5171-815X], Moritz Roidl [0000-0001-7551-9163]
Chair of Material Handling and Warehousing, TU Dortmund University,
Email: larissa.kraemer@tu-dortmund.de, rico.ahlbaeumer@tu-dortmund.de

2022 IEEE International Conference on Blockchain (Blockchain) | 978-1-6654-6104-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/BLOCKCHAIN55522.2022.00046

***Abstract*—As Multi-Agent Systems, Cyber-Physical Production Systems (CPPS) require mechanisms for the allocation of tasks to multiple agents such as workstations or mobile robots. One approach is market-based task allocation (TA) which faces challenges related to fraud and intransparency. Integrating blockchain technology (BCT) into TA enables transparency and tamper resistance. It also allows for the downstream integration of financial transactions in CPPS. This paper proposes a method for two-stage market-based TA in CPPS enabled by BCT. A formal definition for a reverse first-price-sealed-bid auction including the payment process is presented for the blockchain-based two-stage auctioning in a CPPS. The proposed method is implemented in a prototype and tested with a focus on scalability. Transaction times and costs are considered within the experiments. The evaluation reveals that the proposed process is functionable and that financial transactions can be completely automated within CPPS. The time and gas usage of transactions scale in most scenarios and only take a few seconds for completion.**

***Index Terms*—Cyber-Physical Production System; Multi-Agent System; Task Allocation; Blockchain; Auction**

I. Introduction

Digitization and increasing customer demands have led to the emergence of Cyber-Physical Production Systems (CPPS). These systems enable automated, self-configuring and robust production by merging the physical and digital world [1], [2]. In a CPPS stakeholders such as vendors, lessors and customers interact with each other, either directly by placing orders or providing material or indirectly by owning agents. These stakeholders are interested in transparent and tamper-proof processes, especially regarding prices and payment. CPPS often comprise a multitude of heterogeneous agents such as mobile robots and carts which do not necessarily belong to one company. The coordination of these agents combined with the management of incoming tasks results in high levels of complexity and challenges regarding robustness against failures and manipulations as well as data security [3]. Task allocation (TA) plays an essential role in this. A common approach to this is using decentralized market-based TA [4]. To guarantee transparent and tamper-proof negotiations, transactions and documentation, blockchain technology (BCT) is one approach to the solution [5]–[7].

The use of BCT also builds the basis for automated financial transactions. BCT promises a reliable, transparent TA for CPPS considering the multitude of agent classes. Hence, the underlying research question of this contribution is how to design such a blockchain-based TA mechanism for a two-stage negotiation process in CPPS and which impact regarding time and costs results from this. In Section II, we derive the state of the art of CPPS, TA, BCT and its intersections. Then we design a blockchain-based two-stage TA for CPPS in Section III which we demonstrate and evaluate with a protoype in Section IV. Finally, we communicate the limitations and design implications in Section V.

II. State of the art

A. Cyber-Physical Production Systems (CPPS)

In Cyber-Physical Systems (CPS), physical and virtual entities interact with each other via embedded hard- and software [1]. Transferred to manufacturing, CPPS that enable automated, flexible and self-configuring production have emerged [2]. CPPS provide several benefits regarding the handling of an increasing production variety, short delivery times and unforeseen events. They also promise a scalable production [8]. In CPPS, agents such as machines, smart bins and mobile robots interact with each other. Agents in this context are entities that collect data from the environment or other agents to build up knowledge [9]. The multitude of heterogeneous agents collaborating in CPPS to manufacture products form a Multi-Agent System (MAS) [9].

MAS provide several benefits for production. Agents with the same functionalities can collaborate on tasks to solve them faster, more efficiently or cheaper. Additionally, the multitude of agents improves the robustness of such systems against failures or malfunctions [4]. However, in MAS it has to be determined which agent solves which task. In a scalable scenario, this becomes increasingly complex for centralized solutions [10].

B. Task Allocation (TA) in Multi-Agent Systems (MAS)

Depending on the particular system and environmental conditions, TA algorithms in MAS can have different objectives such as the minimization of waiting times for deployed agents, the maximization of the total throughput or the minimization of costs, e.g. minimization of driving distance or energy consumption of involved agents [10].

For TA in MAS, centralized and decentralized approaches can be distinguished [4]. In centralized approaches, one agent

978-1-6654-6104-7/22/$31.00 ©2022 IEEE
DOI 10.1109/Blockchain55522.2022.00046

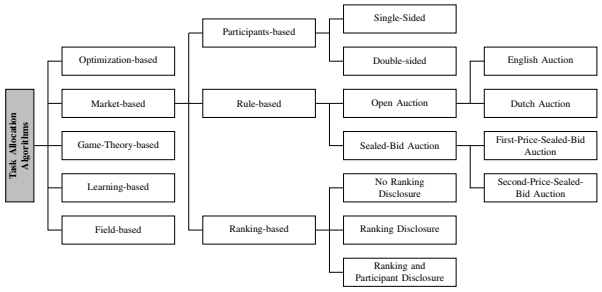

Fig. 1. TA algorithms with a focus on market-based TA based on [7]

acts as a central coordinator which communicates with the other agents and manages the TA. In decentralized approaches, agents negotiate directly with each other taking information about their local environment into account, and find locally optimal solutions for tasks. Based on [11] a task is defined as a subgoal required for an overall goal and independent from other subgoals.

An exemplary case of TA is Multi-Robot Task Allocation (MRTA). Herein, tasks can be distinguished by defining the number of robots required to fulfill the tasks. Single-Robot tasks (SR) need exactly one robot, whereas Multi-Robot tasks (MR) require more than one robot [11]. Considering the context of time and information about the robots, tasks and environment, the task assignment can be classified as Instantaneous Assignment (IA) and Time-Extended Assignment (TEA) [11]. Further distinction can be made between static and dynamic TA. In static TA a re-assignment of tasks is not possible, whereas dynamic allocation allows for the re-assignment of tasks if there is another robot better suited for the task than the one initially chosen [10]. Furthermore, Single-Task robots (ST) can perform only one task at a time, whereas Multi-Task robots (MT) are capable of simultaneously executing multiple tasks [11].

For TA several approaches with different types of algorithms are applicable [4], [10]. The most relevant ones according to [4], optimization-based and market-based approaches, are presented in the following. In addition, game-theory-based, learning-based and field-based approaches are discussed in literature [4].

The **optimization-based** approach uses exact algorithms with dynamic programming or Mixed Integer Linear Programming (MILP). These approaches consider all parameters which are to be minimized or maximized and all globally available information and known restrictions. This is an NP-hard problem and therefore infeasible for large and complex system. Hence, heuristics, meta-heuristics or hyper-heuristics are applied instead of exact algorithms. [10]

A second approach is called **market-based** or auction-based and is oriented towards economic principles. The agents act as auctioneers or bidders and the auctions take place based on locally available information [10]. An agent can act as a central auctioneer or the auctions can be performed in a distributed manner by the various agents in MAS [4]. The market-based approach is scalable, flexible, robust and new tasks can be easily integrated. However, it is not guaranteed to reach the global optimum of optimization-based solutions. In [7] the following taxonomy for auction types is suggested: Firstly, there are participant-based auctions which can be single-sided or double-sided. While single-sided auctions either have one seller and multiple buyers (forward auction) or one buyer and multiple sellers (reverse auction), double-sided auctions include multiple sellers and buyers [7]. The second type is rule-based bidding which can be performed as an open auction or a sealed-bid auction. In an open auction, e.g. English or Dutch auctions, bidders are aware of other bids and are allowed to bid more than once. In a sealed-bid auction, bidders only bid once and bids are kept private. In a first-price-sealed-bid auction (FPSBA), the winner pays their own bidding price, while in a second-price-sealed-bid auction (SPSBA), also called Vickrey auction, the winner pays the second-highest price. Either form of rule-based bidding can be executed as a forward or reverse auction [7]. Thirdly, there are ranking-based auctions with either no ranking disclosure, ranking disclosure or ranking and participant disclosure [7]. Known algorithms for the market-based approach are listed in [4], [10].

Even though market-based approaches are popular in literature, they bring some challenges with them. In a centralized approach, a trusted third party manages the auctions. The bidders have to trust in the sincerity of this party. A decentralized approach minimizes this risk, but agents could still misbehave by modifying their own bids or bids of other bidders. Besides, the auction history might not be public to all participating agents and might not be verifiable [6]. Trust between agents and stakeholders, transparency and accountability are other major challenges in cross-company MAS [12]. One approach to improve these critical points in market-based TA in MAS is the use of BCT.

C. Blockchain Technology (BCT) - Technical Details

BCT is a distributed and practically immutable database maintained by a decentralized network. Validation is achieved by a consensus mechanism and by back-referencing of blocks [13]. For market-based auctions, BCT enables a trust frontier through its immutability and decentralized consensus without an intermediate trusted third party. In addition, BCT allows for smart contracts, which are pieces of code autonomously executed based on predefined rules [14]. Smart contracts promise to also increase the trust for auction-based approaches as they are stored on-chain and are therefore immutable.

Only some of the multitude of existing BCT frameworks are able to include smart contracts and can therefore be used for TA, for example Ethereum [15] or Hyperledger [16]. In this work, we focus on Ethereum due to its convenient use of smart contracts and private testnets, extensive documentation and good community support.

In its mainnet, Ethereum uses a Proof of Work (PoW) consensus called *Ethash*. Following the EIP-225 [17] it is possible to use a Proof of Authority (PoA) consensus mechanism called Clique, which is implemented by the public testnets Rinkeby

and Goerli. It is designed to reduce the number of malicious transactions through the PoW consensus of Ethereum, as the security of this consensus is only as strong as the computing power behind it. The initialization of a Clique testnet [18] is very similar to Ethash as both need a genesis file. This file serves as the first entry in a blockchain and gives specifications to the configuration of the blockchain. Clique utilizes parts of the genesis file of the PoW consensus, e.g. the *nonce* field is used to add and remove signers. This is done due to maintenance overhead when adding additional functionalities for each consensus mechanism.

The creation of a block with Clique is similar to Ethash, but without mining. A miner collects and executes transactions, updates the network state, calculates a hash of the block, and signs the block using their private key. To limit the number of processed transactions in a testnet per signer, Clique allows a signer to create one block per blocktime, which is set to $15\,\mathrm{s}$ to mimic the mainnet. This also serves as a security function to prevent malicious signers.

Transaction costs, i.e. the effort to execute an operation, in Ethereum are comprised of *gas* units and the *gas* price. These costs are paid in Ethereum's currency Ether (*ETH*) and the price for a *gas* unit is denoted in *gwei*. $1\,gwei$ equals $10^{-9}\,ETH$. In Clique the transaction costs are paid directly to the signer, that can then use these fees to pay others involved in this operation. Additionally, as Clique does not have mining, the price of a transaction might not be incidental to the time it takes to approve a transaction. For each transaction, a base fee of $21\,000\,gas$ is set and is increased when interacting with a smart contract [19].

D. Blockchain-based Auctions for Task Allocation in CPPS

Combining auctions with BCT to increase security and privacy has gained recent attention in literature. In [20] an Ethereum-based protocol to achieve bids privacy in sealed-bid auctions is presented. In [21] such a protocol is proposed. A prototype of a blockchain-based SPSBA is demonstrated and evaluated in [22]. [23] also describe an Ethereum-based implementation for sealed-bid auctions and [6] compare four types of auctions implemented in Ethereum regarding cost and time efficiency. [5] propose a blockchain-based reverse English auction with smart contracts to find a suitable trusted infrastructure provider. An overview of the current state of the art in terms of blockchain-based auctions is given in [7]. They also propose a framework for decentralized auctions using Ethereum [7]. Set in the scenario of TA, [24] propose an architecture for dynamic and decentralized TA via BCT in flooding disaster rescue. They use a private blockchain for the coordination of heterogeneous robots and different types of tasks. In [25], blockchain and smart contracts are used to immutably record a history of robot transactions, to integrate new agents into the system and to reconfigure the system on the fly.

Besides these approaches to TA and agent management, several authors also recognize the potential of BCT for CPPS or MAS in general [3], [12], [26]–[28]. In addition to TA, mentioned applications include pay-per-use models [29] or records of registered and authorized devices [30]. The reviewed CPPS publications either stay on a theoretical level such as [26], [28], or demonstrate small-scale, enclosed implementations [27], [31].

TABLE I
SMART CONTRACT FUNCTIONS OF AGENT INTERACTION CLASSES

Function	$\mathcal{A}_1$	$\mathcal{A}_2, \ldots, \mathcal{A}_{m-1}$	$\mathcal{A}_m$
addAuction	x	x	
addBid		x	x
completeAuction	x	x	
payPrice	x	x	
serviceProvided		x	x

As evident from the researched literature, the potential of using BCT for TA in and out of the manufacturing context has been realized. However, as of now, it remains unclear how a multi-stage blockchain-based TA approach in CPPS can look like, how it can be implemented and which implications regarding scalability (time and costs) result from this.

III. BLOCKHAIN-BASED TWO-STAGE TASK ALLOCATION IN CPPS

A. Prelude

The proposed blockchain-based two-stage TA can be classified according to the presented taxonomies in Section II. In the auctioning processes, multiple agents bid on one tender and are instantaneously assigned which equals *SR IA*. Once a task has been assigned it cannot be re-assigned to another agent (static TA) and each agent can only carry out one single task at a time (*ST*).

The TA aims at reducing the costs of the overall system by assigning tasks according to objectives such as low delay time of tasks and low energy consumption for transportation. For this, we use a market-based approach with rule-based bidding, a reverse FPSBA. The winner of an auction is determined by set criteria fixed in a smart contract function prior to the auctioning process. One of these smart contract functions acts as a single, fixed auctioneer. Due to the decentralized nature of BCT, the agents cannot change their bids and trust is gained by a fixed auction history accessible by all agents. The market-based TA takes place in a two-stage process defined as follows:

Definition 1 (Agents Interaction Classes):
Let $\mathbb{A} = \langle \mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m \rangle$ be an ordered set of agent interaction classes. $\beta : \mathbb{A} \times \mathbb{A}$ denotes the *bid* relation and σ the *offer auction* relation. $\forall(\mathcal{A}_i, \mathcal{A}_j) \in \beta : i-1 = j \wedge \forall(\mathcal{A}_i, \mathcal{A}_j) \in \sigma : i+1 = j \Leftrightarrow \forall(\mathcal{A}_i, \mathcal{A}_j) \in \mathbb{A} \times \mathbb{A} : (\mathcal{A}_i, \mathcal{A}_j) \in \beta \Leftrightarrow (\mathcal{A}_j, \mathcal{A}_i) \in \sigma$.

Definition 2 (Agent of an Agent Interaction Class):
Let $\mathcal{A}_i = \{a_1^{\mathcal{A}_i}, a_2^{\mathcal{A}_i}, \ldots\}$ with $i \in \mathbb{N}_{>0}$ be the set of agents in the agent interaction class $\mathcal{A}_i$. Each agent has its own wallet in the blockchain and is able to interact independently with the blockchain via smart contracts.

Definition 3 (Blockchain Rules for an Agent):
Each agent $a^{\mathcal{A}_i}$ has to adhere to the following rules:

1) Each agent interaction class $\mathcal{A}_i$ has its own set of smart contract functions provided in Table I.
2) $a^{\mathcal{A}_i}$ is only permitted to add an auction $\Pi^{\mathcal{A}_i}$ or bid on an auction $\Pi^{\mathcal{A}_{i-1}}$ if it is registered with the smart contract. This prevents unauthorized behaviour.
3) Each auction $\Pi^{\mathcal{A}_i}(a^{\mathcal{A}_i})$ has exactly one owner $a^{\mathcal{A}_i}$. The owner is the agent which created the auction.
4) $a^{\mathcal{A}_i}$ is only permitted to bid once on an agent's auction $\Pi^{\mathcal{A}_{i-1}}$ according to the definition of the bid relation in Definition 1. This ensures that agents are able to fulfill the task of the auction.
5) $a^{\mathcal{A}_i}$ is only permitted to read the content of its own auction $\Pi^{\mathcal{A}_i}(a^{\mathcal{A}_i})$. This ensures the sealed-bid auction, i.e. bidders cannot see bids of other bidders.
6) $a^{\mathcal{A}_i}$ is only permitted to close its own auction $\Pi^{\mathcal{A}_i}(a^{\mathcal{A}_i})$.

B. Process of Task Allocation

Algorithms 1, 2, 4, 3 and 5 show the blockchain process of each agent in pseudocode. The process is additionally described in figure 2. We set m to 3, which results in $\mathbb{A} = \langle \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3 \rangle$, where $\mathcal{A}_1 = \{a_1^{\mathcal{A}_1}\}$, $\mathcal{A}_2 = \{a_1^{\mathcal{A}_2}, a_2^{\mathcal{A}_2}, \ldots, a_l^{\mathcal{A}_2}\}$ and $\mathcal{A}_3 = \{a_1^{\mathcal{A}_3}, a_2^{\mathcal{A}_3} \ldots, a_k^{\mathcal{A}_3}\}$.

Agent $a_1^{\mathcal{A}_1}$ receives a task $t_1 \in T^{\mathcal{A}_1}$ for which the fulfillment of another task $t_2 \in T^{\mathcal{A}_2} \times \mathcal{A}_1$ is obligatory. In the production context, this might be an assembly step (t_1) for which specific material (t_2) is required. $a_1^{\mathcal{A}_1}$ now adds auction $\Pi^{\mathcal{A}_1}$ to the blockchain using `addAuction()` with all relevant input parameters for t_2 to be able to continue with t_1, see Algorithm 1. t_2 can be fulfilled by agents $a^{\mathcal{A}_2}$. In manufacturing, these might be carts at different positions to provide material for assembly. To fulfill t_2, another Task $t_3 \in T^{\mathcal{A}_3} \times \mathcal{A}_2$ can be necessary. For instance, to provide material (t_2), carts are dependent on a matter of transport such as mobile robots (t_3). Thus, each agent $a^{\mathcal{A}_2}$ also adds an auction $\Pi^{\mathcal{A}_2}$ to the blockchain with `addAuction()`, see Algorithm 2. Once these auctions have been published on the blockchain, each agent $a^{\mathcal{A}_3}$ calculates their bid $\rho = b_{t_3}^{a^{\mathcal{A}_3}}(a^{\mathcal{A}_2})$ defined as $b : T^{\mathcal{A}_3} \times \mathcal{A}_2 \times \mathcal{A}_3 \to \mathbb{R}_0^+$. This bid is based on the monetary representation of impact factors such as the driving distance.

The agents bid on the auctions $\Pi^{\mathcal{A}_2}$ with `addBid()`, see Algorithm 3. If enough bids are submitted, the agents $a^{\mathcal{A}_2}$ will close their auctions $\Pi^{\mathcal{A}_2}$ with `completeAuction()`. For each auction, the smart contract will determine the winner $a_\pi^{\mathcal{A}_3}(\Pi^{\mathcal{A}_2})$ with $\pi \in [1, k]$ with its bid $b_{t_3}^{a_\pi^{\mathcal{A}_3}}(a^{\mathcal{A}_2})$, i.e. the agent with the lowest bid. Based on these winning bids, the agents $a^{\mathcal{A}_2}$ calculate their bids $\tau = b_{t_2}^{a^{\mathcal{A}_2}}(b_{t_3}^{a_\pi^{\mathcal{A}_3}}(a^{\mathcal{A}_2}))$ defined as $b : T^{\mathcal{A}_2} \times \mathbb{R}_0^+ \times \mathcal{A}_2 \to \mathbb{R}_0^+$ for the auction of $\Pi^{\mathcal{A}_1}$ and add their bids to the blockchain with `addBid()`. The closure of the auction $\Pi^{\mathcal{A}_1}$ with `completeAuction()` is analogous to the process of closing $\Pi^{\mathcal{A}_2}$ and results in the overall winning agent $a_{\pi^*}^{\mathcal{A}_2}$ and its bid τ^* and the overall winning agent $a_{\pi^*}^{\mathcal{A}_3}$ with its bid ρ^*.

This triggers the payment process, which runs top-down from $a_1^{\mathcal{A}_1}$ to $a_{\pi^*}^{\mathcal{A}_3}$, shown in Algorithm 4 for $a_1^{\mathcal{A}_1}$ to $a_{\pi^*}^{\mathcal{A}_2}$ and 5 for $a_{\pi^*}^{\mathcal{A}_2}$ to $a_{\pi^*}^{\mathcal{A}_3}$. $a_1^{\mathcal{A}_1}$ deposits the winner's price τ^* in a smart contract with `payPrice()`, which holds the amount until task fulfillment. On the one hand, this guarantees $a_{\pi^*}^{\mathcal{A}_2}$ that $a_1^{\mathcal{A}_1}$ can afford its services, while on the other hand, if a task is not fulfilled, the smart contract will release the amount τ^* and send it back to $a_1^{\mathcal{A}_1}$. Once the amount is deposited, $a_{\pi^*}^{\mathcal{A}_2}$ is notified and, in turn, sends the amount ρ^* to the smart contract with `payPrice()`. In the next step, $a_{\pi^*}^{\mathcal{A}_3}$ fulfills its task in cooperation with $a_{\pi^*}^{\mathcal{A}_2}$. In the manufacturing context, for instance, a robot carries a cart with material to a workstation. If this task is fulfilled, the smart contract will release the amount τ^* to $a_{\pi^*}^{\mathcal{A}_2}$ and ρ^* to $a_{\pi^*}^{\mathcal{A}_3}$ with `serviceProvided()`.

Algorithm 1 Auction for $\mathcal{A}_1$

Require: $t_1 \in T^{\mathcal{A}_1}, t_2 \in T^{\mathcal{A}_2} \times \mathcal{A}_1, \Pi^{\mathcal{A}_1}$ auction for t_2, $a_{\pi^*}^{\mathcal{A}_2}$ winning agent, τ^* winning bid, s smart contract

procedure HANDLETASK($t_1, a_1^{\mathcal{A}_1}$)
  $t_2 \leftarrow prerequisite(t_1)$
  $\Pi^{\mathcal{A}_1} \leftarrow addAuction(t_2)$ ▷ triggers Alg. 2
  if $state(\Pi^{\mathcal{A}_1}) = success$ **then**
    $(a_{\pi^*}^{\mathcal{A}_2}, \tau^*) \leftarrow completeAuction(\Pi^{\mathcal{A}_1})$
    $wallet(s) \leftarrow payPrice(a_{\pi^*}^{\mathcal{A}_2}, \tau^*)$ ▷ triggers Alg. 4
    if $state(serviceProvided(t_2)) = success$ **then**
      $runTask(t_1)$
    else
      $wallet(a_1^{\mathcal{A}_1}) \leftarrow receive(\tau^*)$
    end if
  end if
end procedure

Algorithm 2 Auction for $\mathcal{A}_2$

Require: $t_2 \in T^{\mathcal{A}_2} \times \mathcal{A}_1, t_3 \in T^{\mathcal{A}_3} \times \mathcal{A}_2$, $\Pi^{\mathcal{A}_2}$ auction for t_3, $\Pi^{\mathcal{A}_1}$ auction for t_2, $a_\pi^{\mathcal{A}_3}$ agent, ρ, τ bids

procedure HANDLETASK($t_2, a^{\mathcal{A}_2}$)
  $t_3 \leftarrow prerequisite(t_2)$
  $\Pi^{\mathcal{A}_2} \leftarrow addAuction(t_3)$ ▷ triggers Alg. 3
  if $state(\Pi^{\mathcal{A}_2}) = success$ **then**
    $(a_\pi^{\mathcal{A}_3}, \rho) \leftarrow completeAuction(\Pi^{\mathcal{A}_2})$
    $\tau \leftarrow calculateBid(\rho)$
    $\Pi^{\mathcal{A}_1} \leftarrow addBid(\tau)$
  end if
  return $\Pi^{\mathcal{A}_1}$
end procedure

Algorithm 3 Auction for $\mathcal{A}_3$

Require: $t_3 \in T^{\mathcal{A}_3} \times \mathcal{A}_2$, $\Pi^{\mathcal{A}_2}$ auction for t_3, ρ bid

procedure HANDLETASK($t_3, a^{\mathcal{A}_3}$)
  $\rho \leftarrow calculateBid(t_3)$
  $\Pi^{\mathcal{A}_2} \leftarrow addBid(\rho)$
  return $\Pi^{\mathcal{A}_2}$
end procedure

TABLE II
SMART CONTRACT FUCTIONS FOR THE TWO-STAGE TASK ALLOCATION

No.	Function	Input Parameters	Minimum required Gas Units	Emitted Event Data
(I)	`addAuction`	`(privateKey)`	55406	owner, auctionnumber
(II)	`addBid`	`(privateKey, auctionNumber, bid)`	121651	auctionnumber
(III)	`completeAuction`	`(privateKey, auctionNumber)`	55880	auctionnumer, winner, winning bid
(IV)	`payPrice`	`(privateKey, auctionNumber)`	39737	auctionnumber, bidder
(V)	`serviceProvided`	`(privateKey, auctionNumber)`	45354	auctionnumber, owner, bidder

Algorithm 4 Payment process for $\mathcal{A}_2$

Require: $t_2 \in T^{\mathcal{A}_2} \times \mathcal{A}_1, t_3 \in T^{\mathcal{A}_3} \times \mathcal{A}_2$, $\Pi^{\mathcal{A}_2}$ auction for t_3, $\Pi^{\mathcal{A}_1}$ auction for t_2, ρ^* winning bid, s smart contract

procedure PAYAGENT($a_{\pi^*}^{\mathcal{A}_2}$, τ^*)
  $a_{\pi^*}^{\mathcal{A}_3} \leftarrow getWinner(\Pi^{\mathcal{A}_2}(a_{\pi^*}^{\mathcal{A}_2}))$
  $\rho^* \leftarrow getBid(a_{\pi^*}^{\mathcal{A}_3})$
  $wallet(s) \leftarrow payPrice(a_{\pi^*}^{\mathcal{A}_3}, \rho^*)$ ▷ triggers Alg. 5
  if $state(serviceProvided(t_3)) = success$ **then**
    $runTask(t_2)$
    if $serviceProvided(\Pi^{\mathcal{A}_1})$ **then**
      $wallet(a_{\pi^*}^{\mathcal{A}_2}) \leftarrow receive(\tau^*)$
    end if
  else
    $wallet(a_{\pi^*}^{\mathcal{A}_2}) \leftarrow receive(\rho^*)$
  end if
  return $serviceProvided(t_2)$
end procedure

Algorithm 5 Payment process for $\mathcal{A}_3$

Require: $t_3 \in T^{\mathcal{A}_3} \times \mathcal{A}_2$, $\Pi^{\mathcal{A}_2}$ auction for t_3, ρ^* winning bid

procedure PAYAGENT($a_{\pi^*}^{\mathcal{A}_3}$, ρ^*)
  $runTask(t_3)$
  if $serviceProvided(\Pi^{\mathcal{A}_2}(a_{\pi^*}^{\mathcal{A}_2}))$ **then**
    $wallet(a_{\pi^*}^{\mathcal{A}_3}) \leftarrow receive(\rho^*)$
  end if
  return $serviceProvided(t_3)$
end procedure

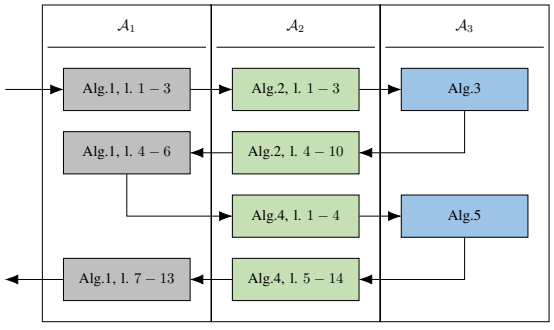

Fig. 2. Visualized sequence of the TA Algorithms 1 - 5

C. Implementation of the Prototype in a CPPS

A prototype of the proposed blockchain-based two-stage TA mechanism is implemented in the manufacturing scenario of a CPPS such as [28], [32]. In the scenario, a drone shall be assembled at a workstation. We focus on a single assembly step, which the workstation receives. For this assembly task, material has to be provided which is stored in carts in a ground storage. The same material can be provided by different carts which require mobile robots for transport. Following the process defined above, this amounts to the following:

- $\mathcal{A}_1$: set of workstations $a^{\mathcal{A}_1}$
- $\mathcal{A}_2$: set of carts $a^{\mathcal{A}_2}$
- $\mathcal{A}_3$: set of robots $a^{\mathcal{A}_3}$

In this context, the TA aims at reducing the costs of the CPPS by assigning tasks according to the minimal driving distance of the robots. The scenario is implemented in the triad of the game engine Unity [33], the blockchain framework Ethereum and the *.NET* integration library Nethereum [34] with a *web3* Frontend. Unity serves as the basis for interaction and provides the visualization of the scenario with a configurable number of carts and robots. Figure 3 shows the example of a scaled scenario in which the robots are bidding on the auctions of the carts. For Ethereum, we use an individually configured private testnet based upon Go Ethereum (Geth) [18]. Clique is used as the consensus mechanism due to security reasons mentioned in Section II-C. The genesis file is configured with recommended settings [18] for Geth and has a gas limit of 1.6×10^9 *gas*.

IV. ANALYSIS AND EVALUATION

A. Design of Experiments

To test the proposed blockchain-based TA mechanism, we concentrate on the smart contract functions shown in Table II with regard to the execution time and related transaction costs. In this context, we define the execution time as the time span from the Unity-request for a blockchain transaction until the receipt from this transaction is received in Unity, i.e. the transaction has been validated in the blockchain. In the following, we set a fixed *gas* price of 1 *gwei* to analyze the amount of *gas* units each transaction requires.

As the TA mechanism is proposed for CPPS which can comprise multiple entities in each agent class and requires scalable solutions, we demonstrate the mechanism for several configurations defined as $(1, l, k)$, with 1 workstation, l carts and k robots. The smallest configuration we test is $(1, 1, 1)$ to demonstrate the general working of the mechanism and as

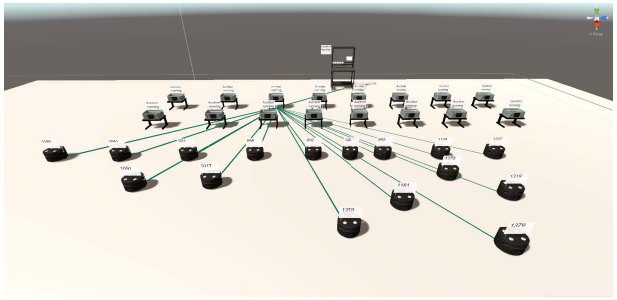

Fig. 3. Exemplary image of the highest-scaled scenario with one workstation, 16 carts and 16 robots

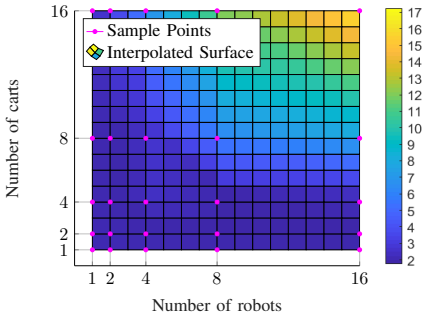

Fig. 4. Time in seconds for `addBid()` of robots depending on the number of involved agents (carts and robots)

a benchmark for higher-scaled scenarios. The most comprehensive scenario we test is $(1, 16, 16)$, with intermediate steps following $l, k = 2^x$ for $x \in [0, 1, 2, 3, 4]$. This results in 25 test scenarios. To determine a suitable number of samples for each scenario, we refer to the Chernoff Bound [35].

Definition 4 (Chernoff Bound for sample complexity [35]): Let $\epsilon \in (0, 1)$ be the accuracy and $(1 - \delta) \in (0, 1)$ be the confidence level, then the Chernoff bound for the sample complexity is defined as follows, where N is the number of needed samples:

$$N \geq \frac{1}{2\epsilon^2} \log \frac{2}{\delta}$$

For our CPPS, we choose $\epsilon = 0.1$ and $\delta = 0.05$ resulting in at least 161 needed samples per experiment. This ensures that with a confidence of 95%, our results do not deviate more than 10% from the result of an infinite number of samples. To ensure the reliability of the results, we choose slightly more than 161 samples, i.e. 200 samples for each scenario. The conducted evaluation includes execution time and used gas for each agent and aggregated for each agent class. The experiments were conducted on a computer running Windows 11 with a *Ryzen 9 5950X*, an *NVIDIA GeForce RTX 3080 Ti*, 64 GB of memory and 1 TB of M2 disk space.

B. Experiments and Evaluation

The evaluation reveals that the minimum required *gas* units for the smart contract functions in Table II stay within the scope of other Ethereum-based smart contract transactions for auctions such as [7], [22]. As stated in Section II, the base feed for each transaction is 21 000 gas. This increases in the case of interaction with the smart contract. Thus, the functions `payPrice()` and `serviceProvided()` are the closest to the base fee as they provide simple transactions with additional input parameters. Functions that have more input parameters, store values in the blockchain or execute calculations such as `addAuction()`, `addBid()` and `completeAuction()` require more gas. This is comparable to the gas consumptions measured in [7] who implement an Ethereum-based online auction. Their consumed *gas* units differ between 30 195 gas for adding a bid and 65 271 gas for evaluating suitable sellers. [22] evaluate gas costs for a Vickrey

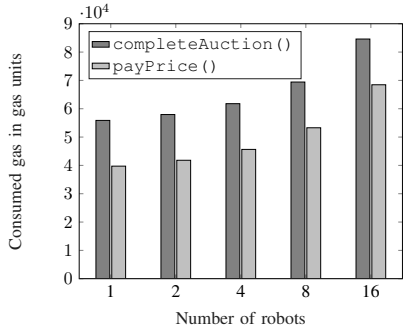

Fig. 5. Used *gas* units for `completeAuction()` and `payPrice()` of carts depending on the number of involved robots

auction and measure between 20 370 gas for withdrawal from an auction and 188 201 gas for starting an auction. As in each of these contributions, the scope of a smart contract function such as `addAuction()` is defined differently, the gas consumption differs. However, these contributions show that the gas consumption within our smart contract is well within the scope of other comparable implementations and, thus, realistic.

For the function `addAuction()`, which is used by the workstation and the carts, the execution time is nearly constant at about 1.82 s. Only the maximum number of 16 carts reveals a slight increase in execution time with 4.01 s. The consumed *gas* units are constant for every scenario and all agents of $\mathcal{A}_1$ and $\mathcal{A}_2$ at 55 406 *gas*.

In contrast to this, increasing the number of agents (carts and robots) impacts the function `addBid()`. While the execution time for adding a bid of a cart is nearly constant at 1.91 s, the execution times from robots `addBids()` show significant scaling effects. Figure 4 displays this effect for the scaling of robots and carts. With either scaling, the execution time increases significantly from 1.92 s to 17.25 s. To understand this, the difference between carts and robots in our experiment has to be considered. While carts only bid once, i.e. on the auction of the workstation, each robot bids several times, i.e. on each auction of each cart. This means, that in a (1,16,16) scenario, each cart bids once (16 bids of carts in total), while each robot bids 16 times, once for the auction of each cart

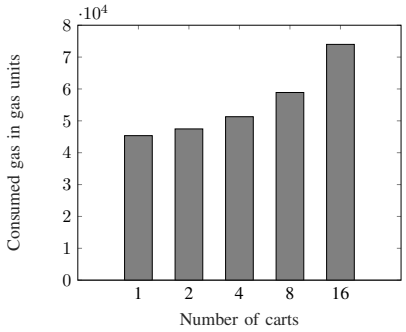

Fig. 6. Used *gas* units for `serviceProvided()` of carts depending on the number of involved carts

(256 bids in total). Hence, the scaling effects are much more visible in bids of robots than in bids of carts.

The function `completeAuction()` also shows significant scaling effects. While the execution time only increases slightly from $1.88\,\mathrm{s}$ to $2.39\,\mathrm{s}$ for carts and from $1.88\,\mathrm{s}$ to $1.96\,\mathrm{s}$ for the workstation, the number of consumed *gas* unity scales linearly with the number of incoming bids starting at $55\,880$ *gas*. This is noticeable in the functions triggered by the workstation as well as the carts. While for the workstation, the consumed *gas* units increase with the number of bids from carts, for the carts the *gas* units increase with the number of bids from robots as displayed in Figure 5. This effect results from the design of the smart contract function which compares all accepted offers and chooses the cheapest one.

The effect of an increasing number of *gas* units parallel to more included agents can also be seen in the `payPrice()`-function as Figure 5 shows. While the execution time stays constant at $1.94\,\mathrm{s}$ for the workstation and the carts, the number of *gas* units linearly increases with the number of saved bids in the auction. Thus, the number of consumed *gas* units of the workstation increases with the number of bids received from carts, starting at $39\,737$ *gas* and increasing to $68\,325$ *gas*, while the number of consumed *gas* units of the carts increases with the number of bids from robots, starting at $39\,737$ *gas* and increasing to $68\,419$ *gas*. This shows that the transaction costs for paying the price of the winner's bid to the smart contract are dependent on the number of bids included in an auction.

For the function `serviceProvided()` we observe a slightly similar dependency. While the execution time of the function remains constant at $1.94\,\mathrm{s}$ for the carts and $1.74\,\mathrm{s}$ for the robots independent of the level of scalability, the consumed *gas* units increase significantly. For the carts, the consumed *gas* units increase linearly with the number of carts, i.e. the auctions and bids of carts, as displayed in Figure 6. The lowest number of consumed *gas* units is $45\,354$ *gas*, increasing to $73\,953$ *gas*. The same effect appears when robots call the function. In this case, the consumed *gas* units scale linearly with the number of robots in the system, starting at $45\,354$ *gas* and increasing to $74\,036$ *gas*. This effect can probably be attributed to accessing the auction which includes all previous bids.

The evaluation proves that the proposed blockchain-based two-stage TA for CPPS is fully operative. With the used Ethereum framework, the execution time increases slowly with the number of involved agents. The transaction costs, i.e. *gas* units (considering a fixed *gas* price) either stay constant or scale linearly with the number of agents in the CPPS.

C. Limitations

The experiments show that the proposed method works in the presented scenario and that the method itself is scalable. However, the presented findings are subject to some limitations. First, we developed and tested the prototype with the Ethereum Framework which in itself has a few limitations such as the limited number of 15 to 45 transactions per second (TPS) and relatively high transaction costs. Hence, in large systems with lots of agents, the called smart contract functions can overlap by several seconds. For instance, this will happen, if the auction of one cart fits into a block in the blockchain, while another auction only fits into the next block. Thus, this auction can be published earlier and bidding can begin while other agents are still adding auctions to the blockchain. This explains slight deviations in the simulated values. In addition, the deviations are expected with regard to the chosen values of the Chernoff Bound, especially because of $\epsilon = 0.1$. Nearly all deviations stay in this 10% range. Also linked to the limiations of Ethereum are the scalability issues our evaluation shows. As it is planned to evolve Ethereum [36], the consensus mechanism will change to Proof of Stake (PoS) allowing for far more TPS with lower transaction costs. Thus, we do not consider this a problem in the future. The method, however, has not yet been tested in a very large scenario with hundreds of agents or in a real-world environment. Additionally, as of now, the handling of events and failures such as agents which are not available or agents which do not give truthful bids has not been carried out so far. Another aspect to be considered is the possible heterogeneity of agents within one agent interaction class such as agents with different properties.

V. Summary and Future Work

This contribution focuses on blockchain-based auctioning for TA in CPPS to improve process transparency and data integrity in such systems. The concept for a blockchain-based two-stage TA method is proposed and demonstrated for CPPS at the hand of a prototype. The evaluation reveals that the method itself is operative and that by using it, negotiation processes and financial flows within CPPS can be reliably automated. However, as the Section IV-C shows, the concept and prototype are subject to several limitations. Hence in our future work, we aim at overcoming those by extending the proposed mechanism to enable failure handling such as non-functionable or ill-willing agents. Since we tested the prototype with a maximum of 33 agents in three agent classes, we plan to extend the proposed concept to more than three classes and a multitude of agents to ensure scalability.

We also plan to establish the proposed mechanism not only in a simulation model, but in a physical environment such

as a research laboratory to examine the necessary (physical) features agents must have to take part in such a blockchain-based CPPS. Additionally, other frameworks or, in the future, the further developed version of Ethereum can be used for further experiments regarding scalability, validation time and transaction costs. Further potential for blockchain-based TA in CPPS lies in the realization of reputation mechanisms for the negotiations.

ACKNOWLEDGMENTS

This research was provided by the 'Blockchain Europe Initiative' funded by the Ministry of Economic Affairs Innovation, Digitalization and Energy of the State of North Rhine-Westphalia (MWIDE) (Grant No. 005- 2003-0068). The work presented in this paper was funded by the 'Deutsche Forschungsgemeinschaft (DFG) - 276879186/GRK2193'.

REFERENCES

[1] acatech, "Cyber-Physical Systems: Driving force for innovation in mobility, health, energy and production." Springer, Berlin, 2011.

[2] L. Monostori, "Cyber-physical Production Systems: Roots, Expectations and R&D Challenges," *Procedia CIRP*, vol. 17, pp. 9–13, 2014.

[3] J. Lee, M. Azamfar, and J. Singh, "A Blockchain Enabled Cyber-Physical System Architecture for Industry 4.0 Manufacturing Systems," *Manufacturing Letters*, vol. 20, pp. 34–39, May 2019.

[4] G. M. Skaltsis, H.-S. Shin, and A. Tsourdos, "A survey of task allocation techniques in MAS," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. Athens, Greece: IEEE, Jun. 2021, pp. 488–497.

[5] M. F. Franco, E. J. Scheid, L. Z. Granville, and B. Stiller, "BRAIN: Blockchain-based Reverse Auction for Infrastructure Supply in Virtual Network Functions-as-a -Service," in *2019 IFIP Networking Conference (IFIP Networking)*. Warsaw, Poland: IEEE, May 2019, pp. 1–9.

[6] C. Braghin, S. Cimato, E. Damiani, and M. Baronchelli, "Designing Smart-Contract Based Auctions," in *Security with Intelligent Computing and Big-data Services*, ser. Advances in Intelligent Systems and Computing, C.-N. Yang, S.-L. Peng, and L. C. Jain, Eds. Cham: Springer International Publishing, 2020, vol. 895, pp. 54–64.

[7] I. A. Omar, H. R. Hasan, R. Jayaraman, K. Salah, and M. Omar, "Implementing decentralized auctions using blockchain smart contracts," *Technological Forecasting and Social Change*, vol. 168, p. 120786, Jul. 2021.

[8] M. Weyrich and C. Ebert, "Reference Architectures for the Internet of Things," *IEEE Software*, vol. 33, no. 1, pp. 112–116, Jan. 2016.

[9] A. Gjikopulli, "A survey on Multi-Agent Systems (MAS)," *Chair of Network Architectures and Services, Department of Computer Science, Technical University of Munich*, 2020.

[10] M. De Ryck, M. Versteyhe, and F. Debrouwere, "Automated guided vehicle systems, state-of-the-art control algorithms and techniques," *Journal of Manufacturing Systems*, vol. 54, pp. 152–173, Jan. 2020.

[11] B. P. Gerkey and M. J. Matarić, "A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, Sep. 2004.

[12] D. Calvaresi, A. Dubovitskaya, J. P. Calbimonte, K. Taveter, and M. Schumacher, "Multi-Agent Systems and Blockchain: Results from a Systematic Literature Review," in *Advances in Practical Applications of Agents, Multi-Agent Systems, and Complexity: The PAAMS Collection*, ser. Lecture Notes in Computer Science, Y. Demazeau, B. An, J. Bajo, and A. Fernández-Caballero, Eds. Cham: Springer International Publishing, 2018, vol. 10978, pp. 110–126.

[13] N. Kannengießer, S. Lins, T. Dehling, and A. Sunyaev, "What Does Not Fit Can be Made to Fit! Trade-Offs in Distributed Ledger Technology Designs," in *Proceedings of the 52nd Hawaii International Conference on System Sciences*, Jan. 2019, pp. 7069–7078.

[14] F. Glaser, "Pervasive Decentralisation of Digital Infrastructures: A Framework for Blockchain enabled System and Use Case Analysis," in *Proceedings of the 50th Annual Hawaii International Conference on System Sciences*, Jan. 2017, pp. 1543–1552.

[15] E. Foundation, "Ethereum." [Online]. Available: https://ethereum.org

[16] H. Foundation, "Hyperledger." [Online]. Available: https://www.hyperledger.org/about

[17] "Ethereum Improvement Proposals." [Online]. Available: https://eips.ethereum.org/

[18] go-ethereum authors, "Go Ethereum." [Online]. Available: https://geth.ethereum.org/

[19] E. Community, "Ethereum Design-Rationale." [Online]. Available: https://eth.wiki/en/fundamentals/design-rationale

[20] H. S. Galal and A. M. Youssef, "Succinctly Verifiable Sealed-Bid Auction Smart Contract," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, ser. Lecture Notes in Computer Science, J. Garcia-Alfaro, J. Herrera-Joancomartí, G. Livraga, and R. Rios, Eds. Cham: Springer International Publishing, 2018, vol. 11025, pp. 3–19.

[21] E.-O. Blass and F. Kerschbaum, "Strain: A Secure Auction for Blockchains," in *Computer Security*, ser. Lecture Notes in Computer Science, J. Lopez, J. Zhou, and M. Soriano, Eds. Cham: Springer International Publishing, 2018, vol. 11098, pp. 87–110.

[22] H. S. Galal and A. M. Youssef, "Trustee: Full Privacy Preserving Vickrey Auction on Top of Ethereum," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, A. Bracciali, J. Clark, F. Pintore, P. B. Rønne, and M. Sala, Eds. Cham: Springer International Publishing, 2020, vol. 11599, pp. 190–207.

[23] Y.-H. Chen, S.-H. Chen, and I.-C. Lin, "Blockchain based smart contract for bidding system," in *2018 IEEE International Conference on Applied System Invention (ICASI)*. Chiba: IEEE, Apr. 2018, pp. 208–211.

[24] T. L. Basegio, R. A. Michelin, A. F. Zorzo, and R. H. Bordini, "A Decentralised Approach to Task Allocation Using Blockchain," in *Engineering Multi-Agent Systems*, A. El Fallah-Seghrouchni, A. Ricci, and T. C. Son, Eds. Cham: Springer International Publishing, 2018, vol. 10738, pp. 75–91.

[25] N. Teslya and A. Smirnov, "Blockchain-based framework for ontology-oriented robots' coalition formation in cyberphysical systems," *MATEC Web of Conferences*, vol. 161, p. 03018, 2018.

[26] M. Y. Afanasev, A. A. Krylova, S. A. Shorokhov, Y. V. Fedosov, and A. S. Sidorenko, "A Design of Cyber-physical Production System Prototype Based on an Ethereum Private Network," in *2018 22nd Conference of Open Innovations Association (FRUCT)*, May 2018, pp. 3–11.

[27] M. Y. Afanasev, Y. V. Fedosov, A. A. Krylova, and S. A. Shorokhov, "An application of blockchain and smart contracts for machine-to-machine communications in cyber-physical production systems," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, May 2018, pp. 13–19.

[28] L. Krämer, R. Ahlbäumer, J. Leveling, P. Detzner, M. Brehler, and M. ten Hompel, "Towards a Concept for Blockchain-based Cyber-Physical Production systems," *Wissenschaftliche Gesellschaft für Technische Logistik*, vol. 2021, no. 17, 2021.

[29] X. Gong, E. Liu, and R. Wang, "Blockchain-Based IoT Application Using Smart Contracts: Case Study of M2M Autonomous Trading," in *2020 5th International Conference on Computer and Communication Systems (ICCCS)*, May 2020, pp. 781–785.

[30] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz, "On blockchain and its integration with IoT. Challenges and opportunities," *Future Generation Computer Systems*, vol. 88, pp. 173–190, Nov. 2018.

[31] A. Angrish, B. Craver, M. Hasan, and B. Starly, "A Case Study for Blockchain in Manufacturing: "FabRec": A Prototype for Peer-to-Peer Network of Manufacturing Nodes," *Procedia Manufacturing*, vol. 26, Apr. 2018.

[32] H. Bayhan, M. Meißner, P. Kaiser, M. Meyer, and M. ten Hompel, "Presentation of a novel real-time production supply concept with cyber-physical systems and efficiency validation by process status indicators," *The International Journal of Advanced Manufacturing Technology*, vol. 108, no. 1-2, pp. 527–537, May 2020.

[33] B. Nicoll and B. Keogh, "The Unity Game Engine and the Circuits of Cultural Software," in *The Unity Game Engine and the Circuits of Cultural Software*. Cham: Springer International Publishing, 2019, pp. 1–21.

[34] Nethereum, "Nethereum." [Online]. Available: https://nethereum.com/nethereum.com/

[35] G. C. Calafiore, F. Dabbene, and R. Tempo, "Research on probabilistic methods for control system design," *Automatica*, vol. 47, no. 7, pp. 1279–1293, Jul. 2011.

[36] E. Foundation, "Ethereum upgrades." [Online]. Available: https://ethereum.org/en/upgrades/#eth2