# A Privacy-Preserving Auction Platform with Public Verifiability for Smart Manufacturing

Thomas Lorünser[a], Florian Wohner and Stephan Krenn[b]
*AIT Austrian Institute of Technology, Vienna, Austria*

Keywords: Multiparty Computation, Verifiable Computing, End-to-End Security, Authenticity.

Abstract: The digitization trend in the manufacturing industry is gaining pace and novel cloud based market places will play an important role in the transformation. However, existing market platforms are centrally organized and can not provide the required level of data privacy and trustworthiness needed for the manufacturing industry. In this work we study the security and privacy aspects for the case of a market platform for outsourcing in manufacturing. We show that the requirements identified together with relevant stakeholder are challenging and sometimes also contradicting on the first sight. To address this challenge we combined different cryptographic building blocks into a novel framework for more secure but transparent decentralized data markets. In particular the framework combines secure multiparty computation with zero-knowledge proof of knowledge methods and blockchain to enable flexible sealed-bid auctions which are also publicly verifiable. For evaluation a proof-of-concept was developed and benchmarking results show that the framework can efficiently address all requirements established.

## 1 INTRODUCTION

Emerging needs for novel secure data spaces are driving current developments in cloud computing and the Internet of Things. One such domain is manufacturing, where the sharing economy is expected to have a significant impact in multiple dimensions, ranging from reduced costs, over increased innovativeness and competitiveness, to considerable environmental benefits. However, mutual distrust and data sovereignty is of special concern in the manufacturing industry, e.g., related to corporate secrets and customer data.

Control over data is still hampered in large infrastructures and the trend to centralization is alarming for a prosper economy. By connecting production facilities to the cloud, many security and compliance approaches relying on a pure contractual basis (service level agreements, SLA) must be reconsidered, especially with respect to the existing oligopoly in the cloud market. A clear understanding of emerging security and privacy issues is needed, and security paradigms based on cryptography rather than SLAs have to be considered in order to guarantee confidentiality and data sovereignty in the cloud.

**Contributions.** In this work, we propose a networked, decentralized architecture for end-to-end verifiable yet privacy-preserving auctions, in order to support future manufacturing clouds and marketplaces. Our platform combines different technologies in a new way to provide adequate protection of business secrets as well as transparency and end-to-end verifiability.

On the technical side, this is achieved by a combination of secure multi-party computation and zero-knowledge proofs of knowledge that also incorporates the edge of the network to achieve the main properties envisaged by the different stakeholders. The platform not only allows for determining the lowest bid, but also advanced price-finding mechanisms based on price-ratio methods. Our solution minimizes the necessary mutual trust, not only between different bidders but also towards the auctioneer. We implemented a proof of concept of our approach, integrating and extending a number of existing tools and frameworks, e.g., for MPC and zkSNARKs. The performed benchmarking demonstrates the practicality for realistic sets of parameters for use cases encountered in our use case.

**Outline.** The remainder of the paper is structured in the following way. We provide a detailed overview of

[a] https://orcid.org/0000-0002-1829-4882
[b] https://orcid.org/0000-0003-2835-9093



Lorünser, T., Wohner, F. and Krenn, S.
A Privacy-Preserving Auction Platform with Public Verifiability for Smart Manufacturing.
DOI: 10.5220/0011006700003120
In *Proceedings of the 8th International Conference on Information Systems Security and Privacy (ICISSP 2022)*, pages 637-647
ISBN: 978-989-758-553-1; ISSN: 2184-4356

related work and the state of the art in Section 2. In Section 3 we present the new architecture and explain how it addresses the requirements of our use case. Our framework is then described in Section 4. The evaluation results of our proof-of-concept implementation are reported in Section 5. Finally, we briefly conclude and discuss possible future extensions in Section 6.

## 2 RELATED WORK

MPC can be considered the most practical approach for generic computation on encrypted data. This means that virtually any function can be computed in an MPC system in principle, however, due to the overhead introduced by MPC protocols they are slower than a classical computer by orders of magnitudes. Nevertheless, the first realization of a real application was demonstrated in an auction held for the Danish sugar beets market in 2009 (Bogetoft and et al., 2009). This was the first large-scale and practical application of multiparty computation and enabled farmers to get a fair market clearing price. They used a local setup with three computers in the same room and ran a semi-honest MPC protocol to calculate the clearing price. About 4000 values for prices were supported at most and 1229 bidders participated in the auction. The inputs from the individual bidders were encoded by verifiable secret sharing and the computation lasted for half an hour.

The basic problem of this setting is the lack of scalability of the MPC protocols itself. The chosen setup with 3 nodes prevents the bidders from directly participating in the computation but requires them to first encode their inputs for the MPC nodes. The improved security in this setting is evident, but to further increase the trustworthiness of the system, and to decrease the trust assumptions in the MPC nodes, some form of public verifiability would be desirable.

To cope with this issue new research combined the mechanisms with blockchain and (non-interactive) zero-knowledge proof (NIZK) techniques. The blockchain is an ideal candidate to be used for storage of relevant audit data in an accessible manner, however, because all data written to the blockchain is visible to every party additional machinery is required to maintain confidentiality and privacy. NIZKs enable parties to publish proofs about statements without revealing secrets per se (witnesses) and are therefore an ideal tool to integrate blockchain with the confidentiality preserving MPC functionality.

Sánchez (Sánchez, 2017) proposed Raziel, a system that combines MPC and NIZK to guarantee the privacy, correctness and verifiability of smart contracts. The idea underlying Raziel is a smart contract which in addition to the standard properties also guarantees correctness of auctions. The validity of the generated NIZKs can also be verified by third parties, thereby achieving publicly verifiability.

Another approach to verifiable auctions has been presented in (Galal and Youssef, 2018) and a software prototype can be found on GitHub[1]. The solution combined homomorphic commitments and NIZK together with a verifiable comparison protocol to achieve a secure FPSBA. The system is verifiable and privacy preserving against outsiders, however, a trusted auctioneer is still required because he learns all bids.

Furthermore, Blass and Kerschbaum (Blass and Kerschbaum, 2018) presented Strain, a protocol to implement sealed-bid auctions on top of blockchains that protects the bid privacy against fully malicious parties. In a nutshell, the protocol works as follows: bids are encrypted bitwise and are stored on the blockchain. Bidders then to run interactive zero-knowledge protocols generating proofs of relations between bids, thereby realizing auctions in a peer-to-peer fashion. Albeit being scalable by the peer-to-peer nature, the protocol still needs a semi-trusted auctioneer as arbiter and requires all parties to be online during all auction phases. It also leaks the full order of all bids compared to the winning bid as required by auctions.

Kosba et al. (Kosba and et al., 2016) presented Hawk, a framework to establish privacy preserving smart contracts on the Ethereum blockchain. Hawk is intended to protect transaction data on chain by leveraging zero-knowledge proof techniques. The goal was an easy-to-use framework providing a compiler managing the cryptographic tasks. Up to our knowledge, the Hawk framework has still not been released yet. However, it is considered that the approach cannot be efficiently integrated with MPC.

In another work (Galal and Youssef, 2019), Galal and Youssef utilized Zero-Knowledge Succinct Non-interactive Argument of Knowledge (zk-SNARK) to realize privacy friendly auctions on a blockchain. However, their solution makes use of a trusted auctioneer who learns the bids. This is contrary to our goals; however, the approach contains interesting aspects and by realizing the auctioneer in a distributed fashion using MPC, the concept resembles the core ideas of our approach. Additionally, cryptocurrencies have been used to incentivize fairness and correctness, and avoid deviations from the MPC or NIZK protocol. In these systems money has to be escrowed in deposits

[1] https://github.com/HSG88/AuctionContract

which are only returned if the behave honestly. This in effect encourages parties to strictly follow the protocols to avoid the financial penalty. Protocols in this direction have been proposed in (Andrychowicz and et al., 2014; Bentov and Kumaresan, 2014; Kumaresan et al., 2016; Kumaresan and Bentov, 2016).

Baum (Baum et al., 2014) proposed publicly auditable MPC; however, this work is mainly of theoretical interest and never really implemented. It also integrates with SPDZ but is likely to be too expensive for practical applications, as the idea is basically to make each computational step verifiable by adding a zero-knowledge proof.

# 3 MARKET PLATFORM

A key requirement in a cloud manufacturing is to optimally match supply and demand, i.e., available manufacturing resources and customers' needs. In the following we thus present a verifiable and privacy-preserving marketplace for manufacturing resources. In our scenario, we consider a marketplace provider, at which owners of manufacturing sites can sign up as *producers* and register their machines as well as meta information such as configurations, quality levels, etc, cf. also Figure 1. *Customers* (or *buyers*) can now put orders, and producers can provide bids to win the order. Leveraging multi-party computation to ensure confidentiality, blockchain to immutably store encrypted bids and results, and zero-knowledge proofs to ensure integrity and verifiability, the marketplace will then match the bids against the order, and announce the winning bid. The precise data flow will be described in Section 4.

## 3.1 Security Requirements

In the following, we introduce the security and privacy requirements to a marketplace for cloud manufacturing applications, which were derived in collaboration with industry partners. On a high level, we found an environment which is aware of their assets but is currently not sure how they could leverage the data they own because of concerns regarding the insufficient protection of business critical information, and the risk of a potentially colluding harmful environment during auctions. More precisely, the requirements are as follows.

**Confidentiality.** Confidentiality of the producers' bids is of utmost importance through all phases of the auction. In particular, the bids do not only need to be protected from unauthorized access through competitors, but also from the platform provider. This is because of the risk of this central entity colluding with certain producers, thereby fully undermining the price finding mechanisms.

Furthermore, our interviews with industry also showed, that production and shop floor data can be business critical. That is, competitors should not gain any information about a producer's current capacity utilization, machinery status, or process information on production lines.

**Integrity.** Besides the requirement of correctness in the case of exclusively honest entities, it is necessary that the integrity of an auction's result can also be guaranteed in the case of a malicious operator of the marketplace. This even needs to hold in the case that the provider is colluding with other entities in the system, including producers and buyers, in order to ensure that no party can manipulate the outcome of the auction in their own interest.

**Availability.** While this is often not considered in the design of cryptographic protocols, it turned out to be of high importance to our partners. On the one hand, producers demand assurance that they will not miss opportunities. On the other hand, related to integrity, producers also need to be guaranteed that they cannot be excluded from an auction; that is, whenever a producer places an offer, it shall also be guaranteed that this offer was indeed considered.

**Anonymity and Pseudonymity.** In addition to confidentiality of bids, producers may wish to even hide the information whether or not they placed an offer for a given auction, as this might already reveal sensitive information about the current utilization or condition of a production line. Depending on the specific business model of the marketplace, this requirement, however, needs to be balanced against the marketplace provider's needs.

**Fairness.** Another important aspect for the clients was fairness in terms of fair conditions. This can also be interpreted as an open and transparent way of computation and selection of the winning offer. However, the evaluation criteria for matching and ranking need to be clear and traceable. The goal is to establish a fair comparison of offers on an comprehensible algorithm to establish fair market prices.

**Transparency.** Finally, transparency requires that all participants in the system are able to trace progress and activities on a high level. Our partners were also interested in historical data in case they were offline for certain times and could not participate in auctions. However, all this functionality needs to be achieved without compromising any of the previous goals, especially those related to confidentiality and privacy.

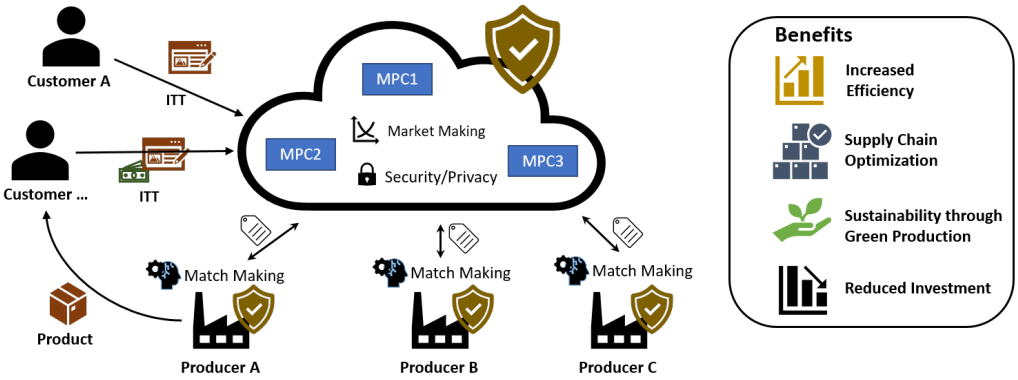


Figure 1: Use case of cloud manufacturing and marketplace.

## 3.2 Auction Mechanism

Auctions are considered a good mechanism to achieve fair market prices for goods. The underlying idea is that every bidder in an auction bids the real value which leads to a real market price. This market mechanism can be undermined in various ways by malicious parties, especially in centralized online platforms. As summarized in (Galal and Youssef, 2018) there are four main types of auctions which are of practical interest and two of which work on hidden bids.

First-price sealed-bid auctions (FPSBA), where bidders submit their bids in sealed envelopes to the auctioneer, which opens them to determine the bidder with the highest bid. Second-price sealed-bid auctions (Vickrey auctions) are similar to FPSBA with the exception that the winner pays the second highest bid instead.

Additionally to the traditional approaches various other price finding mechanisms are relevant in the industry. In the requirements assessment for a manufacturing marketplace, it became clear that it is not only the price which is relevant for selection of the best offer, but also other parameters. Currently delivery options, logistic costs, quality requirements and other parameters are also part of the decision process to select the best offer.

In that sense, it is essential to have a very flexible mechanism for ranking offers. At the heart of our matching mechanism is thus the idea of a *matching score*, which allows to address additional targets and be more versatile in configuration. In this mechanism, the buyer can define different criteria and priorities among them, which should be taken into consideration. This starts from basic capabilities of the production facilities and can be arbitrarily extended for additional topics as mentioned above. These parameters are defined as part of the tender and distributed along with it. The matching is then based on capabilities of a producer, which are immutably registered on a blockchain when signing up as producers or registering equipment. In essence, this leads to a case where the price has to be combined with a matching score to rank results.

Specifically interesting are price finding methods based on linear interpolation. The so called *price ratio methods* are often used to compare offers which are not comparable otherwise. In this approach, a score $R$ for a given bid is computed as:

$$R = \omega_L \cdot \frac{L}{L_{max}} + \omega_P \cdot \frac{P_{min}}{P}, \quad (1)$$

where for simplicity we only consider a single criterion here. In this equation, $L$ means the level of fulfillment of this criterion by an offer, which corresponds to the matching score, and $L_{max}$ is the best level possible. The actual price for a certain offer is then given in $P$, with $P_{min}$ being the lowest in the auction. The $\omega_l$ and $\omega_P$ are weights for the respective ratios on fulfillment grade and pricing, which are defined by the buyer and can be either public and known to the producers as part of the tender or also kept secret, depending on the preferences of the buyer. By inspecting the method it gets clear that not all values can be computed at the edge, i.e., on the client side, as in particular the ratio $\frac{P_{min}}{P}$ of the prices requires the minimum over all offers.

From this analysis it becomes evident that simple price ranking is not enough for winner estimation and the many possible options require a flexible computing system which goes beyond oblivious sorting of bids. In fact, each buyer would like to define his specific matching criteria and also decide on a ranking mechanism to fulfill his needs. This turned out to make the overall system architecture more challenging and rendered many smart contract based approaches from the literature inadequate. On the other hand, the MPC solutions available can cope with the

degree of flexibility required but do not provide means for public verifiability as needed for our architecture.

# 4 FRAMEWORK

The proposed framework is designed to address the security objectives defined in Section 3.1, especially bringing together typically contradicting goals of privacy and verifiability in a single solution. It has a decentralized architecture and follows data minimization principles.

By the use of secure multiparty computation the platform itself is operated in a way that the provider does not learn sensitive data and by making every step of the of the tendering process verifiable the trustworthiness is achieved. For end-to-end verifiability, publicly verifiable zero-knowledge proofs of knowledge are generated for all computations, even for the MPC steps. Finally, to trace all interactions and proofs we use a distributed ledgers which serves as a trust anchor and immutable append-only data base.

## 4.1 Data Flow

In the following we detail the data flow in our platform. To ease understanding, Figure 2 provides a high-level overview, where we omit setup steps for the sake of clarity.

**Setup Phase.** The following setup steps are necessary to operate the system.

1. On the one hand, ZKSetup is used to generate the common reference string (CRS) needed for the NIZKs. On input the security parameter and a circuit, this algorithm outputs the CRS which is assumed to be implicit input to all further algorithms and parties. It is important to note that the system is specifically designed in a way that this step is only needed once and does not need to be invoked again if different ranking mechanisms are used, as they are all supported by the specifically designed circuit with built-in flexibility. In practice this setup algorithm can be run in dedicated setup ceremony, including, e.g., secure hardware elements or dedicated MPC-based ceremonies.
2. On the other hand, RegUser is a protocol which is run by the user and the platform to register with the platform. It is used to generate necessary identities and credentials to authenticate the user and set up the necessary permissions on the ledger.

**Auction Phase.** After the setup is complete the following steps are conducted in the protocol for a particular auction.

1. RegEqm. A client registers equipment (machine) for usage in the system. On input machine parameters, this algorithm outputs a commitment to the machine parameters which is then stored on the ledger. The registration of new equipment has to be done before a producer can participate in an auction.
2. ITT. A buyer sends a request for quotation with relevant parameters to all parties by storing them to the blockchain.
3. Match. Based on the tender information received, the producers compute a matching score for their machines. Based on the score a local matching decision is done to decide whether or not to participate in the auction. If the producer does not participate, the local process is aborted.
4. ComInput. If the producer is participating in the bidding, it computes a NIZK for the matching score and a commitment to the bid. The bid commitment and the proof are stored in the blockchain.
5. Input. In this step the producers (i.e., bidders) send their bids together and matching scores to the MPC system in a secret-shared fashion.
6. CkInput. The MPC system retrieves the corresponding commitments and proofs from the blockchain and verifies them in the encrypted domain. This is done by recomputing the commitments on the shares (for bids and matching scores) at each node and comparing the reconstructed commitments with the plaintext ones. Additionally, each node verifies the proof for the local matching score individually. If either of the checks fails, the system complains about the producer.
7. Compute. The MPC system calculates a ranking based on scores and bids according to the ranking function defined by the buyer.
8. ZKProof. The MPC system generates a NIZK for the winning bid, proving that it is the best ranked result according the predefined ranking function. It does so by each node computing the proof on its share of the result.
9. Reveal. To reveal the result in a verifiable form, the winning bid is reconstructed from the encodings as well as the final proof enabling the verification of the winner calculation. The data is recorded in the blockchain and finalizes a particular auction.

There are many variations possible in practice but this will only result in subtle changes, e.g., if the winning producer and/or bid has to be kept secret.

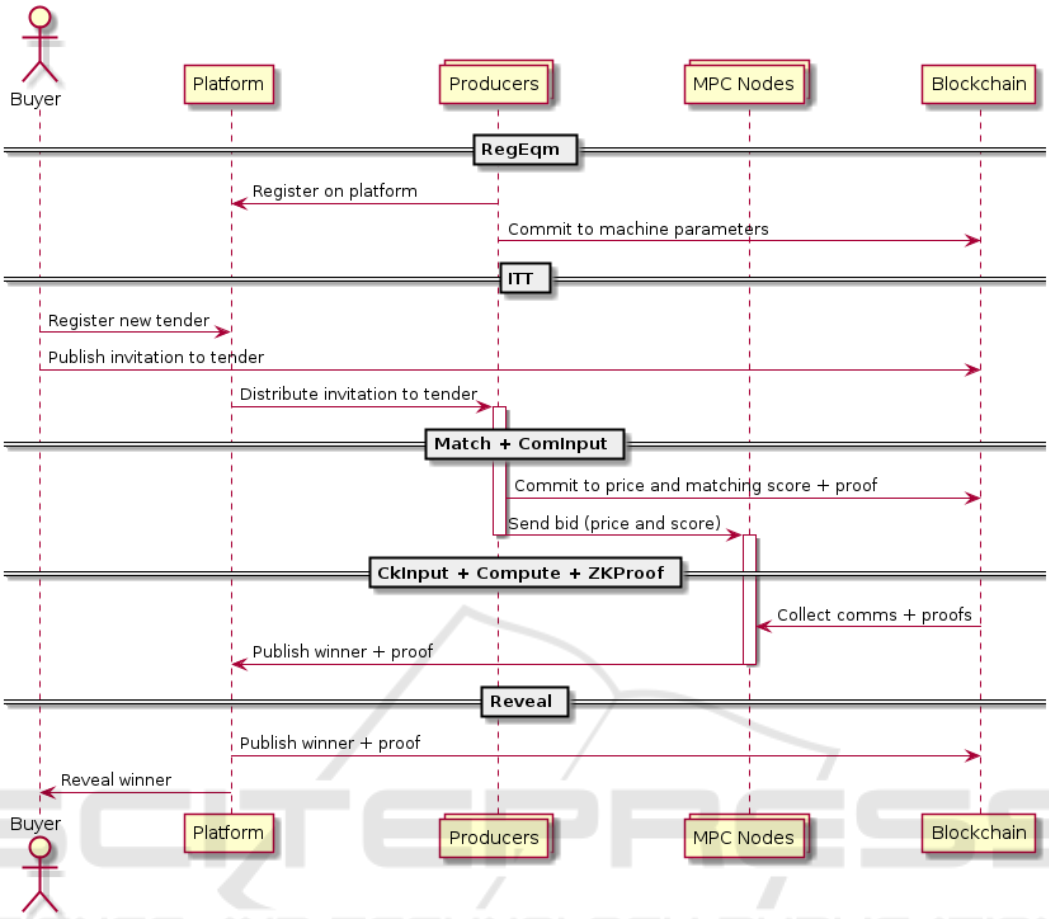


Figure 2: Sequence diagram for an auction.

## 4.2 Protocols

Different protocols have been used, extended and integrated to achieve all desired properties for our framework. At the core we combine multiparty computation with non-interactive zero-knowledge proofs of knowledge (NIZK) to achieve confidentiality and public verifiability that the same time. Regarding MPC we do not rely on any specific protocol but only require a method which is based on secret sharing. However, because we aim at public verifiability the correctness of the computation is going to hold even if all nodes are corrupt. Therefore, depending on the individual assumptions made for the MPC deployment, it can be sufficient to rely on passively secure protocols.

To achieve verifiability, the system is based on adaptive zk-SNARKs as introduced in (Veeningen, 2017). Working with commitments to track different steps in the process is essential to guarantee privacy of sensitive data. However, the protocol is not guaranteeing any authenticity which is essential to track the flow from end to end. Therefore, we leverage ideas from ADSNARK (Backes and et al., 2014) and use signatures on the commitments to assure the authenticity of the data right from the source. In our use case both can be used, standard signatures but also group signatures or delegated signatures (Krenn and Lorünser, 2021), if a certain degree of anonymity is still required, e.g., if it should not be visible which subsidiary of a larger organization the resources belong to.

An important requirement was to reduce the number of times the setup procedures of the zk-SNARKs have to be executed. Ideally, it has only to be done once when initializing the platform, and can then be reused for all subsequent auctions. Therefore, we use case a hybrid approach that turned out to be very efficient. On the one hand we use the idea of subroutines (sub-qaps), which are basically predefined subroutines at setup time but can be connected during proof generation by means of intermediate commitments, to establish the required circuit. This concept is very flexible with only little overhead, i.e., the

additional commitments increase the proof size and verification time for each subroutine defined. To enable even more freedom in the configuration of ranking algorithms we integrate the ideas of universal circuits as presented with MIRAGE (Kosba and et al., 2020). Altogether, at setup time we generate a proof circuitry which comprises both, static elements and freely re-configurable components to get the best of both worlds. Starting from a common pattern of auction markets, we can now instantiate the right building blocks to support a wide range of auction systems with extensive configuration options. To that end, this approach is somehow similar to partially reconfigurable hardware.

## 4.3 Security

In the following we will informally discuss the achieved security goals, a more formal treatment is planned as future work. Moreover, we also give some design rational and explain several architectural decisions.

**Confidentiality.** The privacy of sensitive information is protected in our framework by two main primitives. On the one hand, MPC is used to compute the winner of the auction in a privacy preserving way and sensitive data is protected by the input privacy provided by MPC.

On the other hand, to enable transparency we are recording inputs at different stages in the blockchain. To achieve confidentiality there we use commitments which are also hiding input. Given that sensitive inputs are never handled in cleartext in the system we achieve strong cryptographic protection, which also results in the discussed properties of bid privacy and posterior privacy.

In our marketplace, we even apply a decentralized matching which allows for local computation of a matching score and therefore follows data minimization principles. Alternatives would be a producer-based matching or a platform-side matching, which both would lead to problems: For the former, it would be required to share sensitive information about available capacities and process information with potential competitors, which directly violates the requirements. For the latter, either the platform would learn sensitive information or the matching would have to be done in the encrypted domain.

**Integrity and Correctness.** In essence, the basic idea of the framework is to preserve the integrity and authenticity of data in the system and to prove the correctness of each computation in between. However, instead of directly signing the input to the system, only commitments are signed and also put in a blockchain, which guarantees that producers are bound to their bids and bids.

In our case we use the extractable commitments presented in (Veeningen, 2017). In the original work these particular commitments were used with a dedicated key to distinguish between input from different parties, but this key is produced in the initial setup phase and has to be distributed to parties, which opens up many attack vectors in practical implementations. We rely on locally generated private keys used to sign the commitments, which never leave the local area and are registered with the platform or the blockchain. This would also allow for the use of group signatures for even more flexibility in the management of edge components without sacrificing the security.

After the tender is initiated and all bidders recorded the required data to participate in the auction, the MPC computation is started. The input is comprised of the private bid, the private matching score as well as the data also recorded on the blockchain, i.e., the commitments on initial machine parameters and the matching score, thereby guaranteeing confidentiality while still binding bidders to all input values. Finally, the MPC network not only outputs the winning bid, but also a NIZK proving its optimality, thereby guaranteeing the correctness of the final result.

It is worth noting that the performed local matching introduces another problem with the integrity of data: As the matching score is relevant to calculate the ranking on the platform, it has to be assured that the score was computed correctly. Therefore, the bidder is required to generate a proof on the score before sending it to the MPC system. We do so by forcing the bidder to commit not only to the bid but also to the matching score and additionally to generate a NIZK which is then stored in the blockchain, letting everybody to also verify the local pre-processing.

Finally, it is important to note that with this approach we are basically tweaking the security model of MPC for the overall system. Because the correctness of the computation is publicly verifiable by means of NIZKs, the integrity of the computation can even be assured if all MPC nodes maliciously deviate from the protocol specification. Even more, in our setting malicious behaviour can be attributed to the right stakeholder, i.e., it is not possible to blame the platform for malicious input from bidders or vice versa. This is achieved by letting the MPC system check all inputs for consistency with the information in the blockchain before it computes a result. Only if all inputs are consistent with the stored commitments and the matching score is computed correctly, the MPC system will incorporate the bid in the auc-

tion, and only then it will be able to compute a proof for the winning bid.

As a result, the full auction flow is accompanied with NIZKs and every participant can verify the correctness of the auction from end-to-end. Even if privacy is compromised by an adversary which compromises enough MPC nodes to recover the bids, he will not be able to influence the winning bid or market mechanism.

**Availability.** The availability of the system is provided by a blockchain component which provides the properties to serve as robust and immutable public append only log. Depending on the deployment of the MPC system, also robustness properties such as fairness or guaranteed output delivery can be achieved. Additionally, as the system is non-interactive, client-side computations cannot be interrupted or blocked by individual participants, resulting in a highly available decentralized architecture. Although the platform server is currently needed to run auctions it would also be possible to remove this single point of failure.

**Anonymity and Pseudonymity.** For the given use case it is not desired to build a completely open and permissionless infrastructure. The clients in this system are part of an ecosystem which requires some level of assurance for producers and buyers. Therefore, we only provide pseudonymity for certain steps in the auction. The pseudonyms are maintained at the platform which mainly prevents the buyers from bypassing the business model of the brokerage role of the platform. The only relevant issue for producers might be that one can determine whether or not a specific producer has submitted a bit for a given auction. This leakage can be easily abolished by always participating in the auction protocol but with a $\infty$ bid.

**Fairness.** In our prototype, the tender information was identified as public and could thus be put into the blockchain, which also serves as a broadcast channel in this step, so that all participants are reliably informed about new opportunities as well as the detailed evaluation and ranking criteria of an auction.

Although for certain buyers it would be preferred to not reveal the details of the ranking mechanism (e.g., the weights in Eq. (1)), in all auctions will the matching mechanism be transparent and consistent for all producers.

**Transparency.** By logging every step into the blockchain in a privacy-preserving way and also proving that all computations are correct, we achieve public verifiability. Every user of the system will thus be able to verify all auctions based on the public data stored in the blockchain without compromising the privacy of individual inputs, thereby achieving the requirement of transparency.

# 5 EVALUATION

In order to evaluate the efficiency and practicability of our framework, a proof-of-concept prototype has been implemented in Python, which has been used to study and benchmark different use cases. The basis for the implementation was existing work on *PySNARK*[2], *qaptools*[3] and *MPyC*[4], which have been integrated and extended with novel functionality, notably Universal Circuits. The new framework seamlessly integrates the steps along the data flow.

All measurement results presented in this work have been measured in a local setup on a single Intel NUC computer equipped with an Intel(R) Core(TM) i5-8259U CPU running at 2.30GHz maximum frequency.

**Winning Bid Computation.** The basic operation in winner determination is finding the maximum of the ranked bids. The computation depends on inputs of all bidders as well as the buyer and is therefore done by the MPC system. We present measured computing times for computing the basic result for different numbers of parties without any additional proofs in Table 1. Note that the system we used to take the measurements only had four CPU cores, so not all of the increase in time between the settings with four and five parties is attributable to the inherent computational overhead.

Table 1: Computation time (s) with increasing number of nodes under 3 minutes.

| #bids | 3 parties | 4 parties | 5 parties |
|---|---|---|---|
| 10 | 0.9 | 1.3 | 3.4 |
| 100 | 10.9 | 16 | 35 |
| 1000 | 115.3 | 176.8 | - |

The given measurements hold under the unrealistic assumption of no network latency. We therefore also performed measurements assuming different network delays, as shown in Table 2, which suggests that for our use cases the latency of the network has a higher effect on the overall efficiency than the selected MPC framework, see also (Lorünser and Wohner, 2020).

**Winning Bid Proof Generation.** For our use cases, the generic approach of proving the correctness of ev-

[2]https://github.com/meilof/pysnark

[3]https://github.com/Charterhouse/qaptools

[4]https://github.com/lschoe/mpyc

Table 2: Computation time (s) with increasing network latency between 3 parties.

| #bids | 0 ms | 2 ms | 20 ms |
|---|---|---|---|
| 10 | 0.9 | 1.1 | 4 |
| 100 | 10.9 | 11.4 | 36.6 |
| 1000 | 115.3 | 124.1 | 386.1 |

ery computational step turned out to be unnecessarily inefficient. Alternatively, the size of the equation system used in the zkSNARK, i.e., the quadratic arithmetic program (QAP), can be significantly reduced by generating a single proof at the end of the computation, showing the optimality of the announced winning bid with respect to the defined ranking function.

The proof can be defined as simple as shown in the following listing, where `matching` holds the matching score of all bidders and `prices_rec` are a list of all reciprocals of the prices submitted. The weights defined by the buyer are `w_l` and `w_p`, cf. also Eq. (1). `res` is the output truth value which is 1 if and only if no bid is ranked better than the winner.

```
res = 1
for l, p_rec in zip(matchings,
                    prices_rec):
  score = w_l*l + w_p*p_min*p_rec
  res *= (score >= s_min)
res = auction.get_public_output(res)
```

This type of proof turned out to be very efficient and the measurements in Table 3 show that all steps in proof generation are extremely practical and are even far below the times needed by the MPC system to compute the winner.

Table 3: Performance comparison. Times are in seconds.

| #bids | $t_{Setup}$ | QAP size | $t_{Prove}$ | $t_{verif}$ |
|---|---|---|---|---|
| 10 | 0.4 | 226 | 0.3 | 0.02 |
| 100 | 3.6 | 2206 | 2.1 | 0.6 |
| 1000 | 22.3 | 22006 | 18.7 | 6.6 |
| 10000 | 186 | 220006 | 160 | 65 |

The same performance can be achieved if the winning bid is not the very best but among a predefined threshold or at a particular place, i.e., like in the Vickrey auction. This variation can be achieved by counting the number of bids above a threshold and optionally also below, which results in the same running time as above.

**Fixpoint Operations.** To work with the quotient method as shown above, division and fixpoint operations are needed. Implementing this within the MPC component would have had a significant impact on the performance and drastically limited the number of bids which can be handled in reasonable time. We therefore compute the reciprocal in the pre-processing step at the producer and send it (together with the bid) as input to the MPC system. To ensure end-to-end verification, we then also have to generate a proof that the reciprocal was computed correctly, but this introduces only minimal overhead. In our implementation clients submit the price in integer and fixpoint representation together with the reciprocal and the resulting error term. This results in a very simple and efficient proof computed on the producer side and even less work for the MPC phase, because computation of reciprocals would consume substantial resources. The additional check necessary to prove consistency of fixpoint representation with price value is shown in the listing below, where the `res` is 1 if and only if the multiplication of the price `p_fxp` with the reciprocal `p_rec` corrected by the error term `p_err` equals 1 and the error term given by the representation is within the allowed bound.

```
res = ((p_fxp * p_rec + p_err) == 1)
res *= (s_err * fxpmul_ <= s_fxp)
```

**zkSNARKs.** As discussed in Section 3 a marketplace for outsourcing manufacturing tasks requires a very flexible way to compare offers and match producers. zkSNARKs lack in this respect, because they require the circuits to be proved to be fixed during the setup phase. We circumvent this downside by integrating two mechanisms: subroutines and reconfigurable circuits.

*Subroutines.* On the one hand, we use the properties of adaptive zkSNARKs to also enable subroutines, as already introduced in (Veeningen, 2017). Subroutines are built by committing to their input and output and therefore interconnect these modules to a larger circuit. This approach also provides the first level of reconfiguration and reuse. Existing circuits can be reused without any additional setup call and rewired at runtime by using the according commitments as wiring mechanism to enforce correctness of signals along the computation chain. Figure 3 shows the main structure used in our framework which the blocks being subroutines.

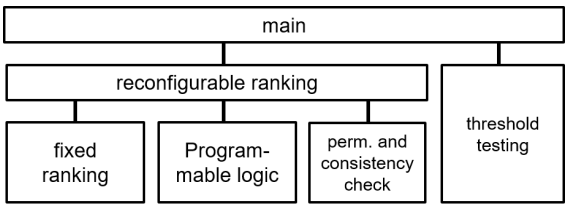


Figure 3: Internal structure of the proof generation.

*Reconfigurable Circuits.* Additionally to the described subroutines we also incorporated universal

circuits. They are general purpose computing blocks which can be reconfigured at runtime without calling setup routines. The concept has been inspired by *MIRAGE* (Kosba and et al., 2020) but integrated with the subroutine mechanism, which is well suited for this. In particular, to assure consistent use of signals within the universal circuit the necessary additional permutation and consistency check was realized as subroutine. A dedicated subroutine was designed which does all checks and was compared to the integrated approach of *MIRAGE*. For our implementation we required only a simplified version of the first opcode presented in the paper, because we did not need any binary operations. The most interesting part four our work was to understand the impact of the size of the universal circuit on the proof generation time and to compare the performance of the original approach in *mirage* to our new way of integrating the permutation and consistency checks.

Permutation and consistency check are performed on public input `li` and private input `zi` which is part of interface commitment for the universal circuit. After calculation of sorted indices `si` the permutation check scores `p1` and `p2` are calculated. Additionally, the consistency checking score `s` is also calculated and both necessary conditions for a successful check are combined into the proof, i.e., assert `p1 = p2` and `s = 0`, which assures intermediary signals connecting gates are consisten, which assures intermediary signals connecting gates are consistent.

```
p1 = 1
p2 = 1
for i in range(len(zi)):
  p1 *= r2 - (zi[i]+r1*l1[i] )
  p2 *= r2 - (zi[si[i]]+r1*l1[si[0]])
res = (p1 == p2)
s = 0
for i in range(1, len(z1)):
  s +=
    (1 - (l1[si[i]] - l1[si[i-1]]))*
    (z1[si[i]] - z1[si[i-1]])
res *= (s == 0)
```

Generally, the use of commitments to intermediary results in circuits increases the size of the proof as well as proofing time. The overhead in size is not an issue for our use case and verification runtime is typically not a problem at all for marketplaces. The advantage of not requiring to call setup phase and generation of additional CRS components is by far more important in real world applications than the increased size or running time experienced. In Table 4 we show the measured times universal circuit composed of different amount of gates, ranging from 10 gates up to 10000 gates. The implemented opcode for our configuration consumes 35 equations in the QAP (20 equations for the logic and 15 equations for the consistency check) if directly combined in one subroutine. The overhead of 21 equations is required for a minimal benchmarking setup.

Table 4: Performance for universal circuit with integrated consistency checks. Times are in seconds.

| #gates | $t_{setup}$ | Qap size | $t_{prove}$ | $t_{verif}$ |
|---|---|---|---|---|
| 10 | 0.5 | 371 | 0.7 | 0.02 |
| 100 | 2.5 | 3521 | 2.2 | 0.02 |
| 1000 | 32 | 35021 | 38 | 0.1 |
| 10000 | 307 | 350021 | 378 | 0.9 |

To further reduce the time for proof generation, we introduced a second approach for the integration of the consistency checking mechanism. In particular we realized the permutation and consistency check as separate subroutines. This means that during configuration of the universal circuit a commitment to the secret values is generated. This commitment is then used together with the public configuration values $l_i$ to perform the permutation and consistency check in a standalone sub-QAP. The measurement results for this variant is shown in Table 5. There can be seen that the size for the logical part can be significantly reduced by about 40% which directly results in reduced proving times. Interestingly, the size of the QAP needed to do permutation and consistency checking is in the same order as the pure uc for our opcode and leads to similar proving times. This is very convenient because it naturally supports parallelization and leads to a reduced overall time although in total about 10% more equations have to be processed.

Table 5: Performance for uc usage with external consistency check. Times are in seconds.

| #gates | uc-Qap | $t_{prove}$ | check-Qap | $t_{prove}$ |
|---|---|---|---|---|
| 100 | 2004 | 0.8 | 1804 | 1.2 |
| 1000 | 20004 | 18 | 18004 | 19 |
| 10000 | 200004 | 160 | 180004 | 180 |

**Overall Complexity.** When considering static scoring functions, the overall complexity of an auction is not dominated by the runtimes given in Table 2 and Table 3. When aiming for a general marketplace supporting flexible scoring functions, also the runtimes presented in Table 5 are to be considered, where the number of gates per bid strongly depends on the complexity of the scoring function and auctioning logic. For the case of a FPSBA with the scoring function defined in Eq. (1), two gates per bid are required.

# 6 CONCLUSION

In this work we showed that simple cryptographically secured sealed bid auctions are not suitable for complex application scenarios like markets places in the manufacturing domain. From the requirements established together with relevant stakeholders, we identified many challenging and partially contradicting objectives which motivated the design of a new architecture and framework. We combined different cryptographic protocols into a framework which can be used to build advanced data markets with a new level of flexibility, security and trustworthiness. The framework enables secure and privacy-preserving price finding for outsourcing tasks in the production industry, but also beyond (Schuetz and et al., 2021). For increased trustworthiness it enables every participant to publicly verify all steps in the auction also in a privacy-friendly way. As core cryptographic tools we combined secure multiparty computation with zero-knowledge proofs of knowledge and enable a seamless experience for the designer of the system. To assess the practical performance we implemented a proof of concept and tested various scenarios. As our main result we were able to show that many requirements given could be achieved with our approach in a single framework and with practical performance. Furthermore, we also showed that the proposed framework is suitable to realize complex use cases in a proof of concept implementation.

In the future, it would be interesting to enable a feedback mechanism for buyers in the form of a rating system, which, however, must not countervail the privacy requirements of producers. Additionally, it would be interesting to see how to extend our concepts to more generic data processing tasks, e.g., statistics or optimization, and how to transfer it to other domains.

# ACKNOWLEDGEMENTS

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 890456 (SlotMachine) and No 830929 (CyberSec4Europe), and the Austrian Research Promotion Agency under the Production of the Future project FlexProd (871395).

# REFERENCES

Andrychowicz, M. and et al. (2014). Secure Multiparty Computations on Bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 443–458. IEEE.

Backes, M. and et al. (2014). ADSNARK: Nearly Practical and Privacy-Preserving Proofs on Authenticated Data. Cryptology ePrint Archive, Report 2014/617.

Baum, C., Damgård, I., and Orlandi, C. (2014). Publicly Auditable Secure Multi-Party Computation. pages 175–196. Springer, Cham.

Bentov, I. and Kumaresan, R. (2014). How to Use Bitcoin to Design Fair Protocols. In *Advances in Cryptology – CRYPTO 2014*, pages 421–439.

Blass, E.-O. and Kerschbaum, F. (2018). Strain: A Secure Auction for Blockchains. In *Computer Security*, pages 87–110. Springer, Cham.

Bogetoft, P. and et al. (2009). Secure Multiparty Computation Goes Live. In *LNCS*, pages 325–343.

Galal, H. S. and Youssef, A. M. (2018). Verifiable Sealed-Bid Auction on the Ethereum Blockchain. Cryptology ePrint Archive, Report 2018/704.

Galal, H. S. and Youssef, A. M. (2019). Trustee: Full Privacy Preserving Vickrey Auction on top of Ethereum. *CoRR*, abs/1905.0.

Kosba, A. and et al. (2020). MIRAGE: Succinct Arguments for Randomized Algorithms with Applications to Universal zk-SNARKs. Cryptology ePrint Archive, Report 2020/278.

Kosba, A. E. and et al. (2016). Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *Security and Privacy, SP 2016*, pages 839–858. IEEE Computer Society.

Krenn, S. and Lorünser, T. (2021). Single-Use Delegatable Signatures Based on Smart Contracts. In Reinhardt, D. and Müller, T., editors, *ARES 2021*, pages 40:1—-40:7. ACM.

Kumaresan, R. and Bentov, I. (2016). Amortizing Secure Computation with Penalties. In *ACM SIGSAC*, CCS '16, pages 418–429, New York, NY, USA. Association for Computing Machinery.

Kumaresan, R., Vaikuntanathan, V., and Vasudevan, P. N. (2016). Improvements to Secure Computation with Penalties. In *ACM SIGSAC, Vienna, Austria, October 24-28, 2016*, pages 406–417.

Lorünser, T. and Wohner, F. (2020). Performance Comparison of Two Generic MPC-frameworks with Symmetric Ciphers. In *SECRYPT 2020*, pages 587–594.

Sánchez, D. C. (2017). Raziel: Private and verifiable smart contracts on blockchains. *IACR Cryptol. ePrint Arch.*, page 878.

Schuetz, C. G. and et al. (2021). A Privacy-Preserving Marketplace for Air Traffic Flow Management Slot Configuration. In *2021 DASC*, pages 1–9.

Veeningen, M. (2017). Pinocchio-Based Adaptive zk-SNARKs and Secure/Correct Adaptive Function Evaluation. pages 21–39. Springer, Cham.