FAST: Fair Auctions via Secret Transactions

Bernardo David[1], Lorenzo Gentile[1(✉)], and Mohsen Pourpouneh[2]

[1] IT University of Copenhagen, Copenhagen, Denmark
bernardo@bmdavid.com, lorg@itu.dk
[2] Copenhagen University, Copenhagen, Denmark
mohsen@ifro.ku.dk

Abstract. Sealed-bid auctions are a common way of allocating an asset among a set of parties but require trusting an auctioneer who analyses the bids and determines the winner. Many privacy-preserving computation protocols for auctions have been proposed to eliminate the need for a trusted third party. However, they lack *fairness*, meaning that the adversary learns the outcome of the auction before honest parties and may choose to make the protocol fail without suffering any consequences. In this work, we propose efficient protocols for both first and second-price sealed-bid auctions with fairness against rational adversaries, leveraging *secret* cryptocurrency transactions and public smart contracts. In our approach, the bidders jointly compute the winner of the auction while preserving the privacy of losing bids and ensuring that cheaters are financially punished by losing a *secret* collateral deposit. We guarantee that it is never profitable for rational adversaries to cheat by making the deposit equal to the bid plus the cost of running the protocol, *i.e.*, once a party commits to a bid, it is guaranteed that it has the funds and it cannot walk away from the protocol without forfeiting the bid. Moreover, our protocols ensure that the winner is determined and the auction payments are completed even if the adversary misbehaves so that it cannot force the protocol to fail and then rejoin the auction with an adjusted bid. In comparison to the state-of-the-art, our constructions are both more efficient and furthermore achieve stronger security properties, *i.e.*, fairness. Interestingly, we show how the second-price can be computed with a minimal increase of the complexity of the simpler first-price case. Moreover, in case there is no cheating, only collateral deposit and refund transactions must be sent to the smart contract, significantly saving on-chain storage.

Keywords: Cryptographic Protocols · Multiparty Computation · Financial Cryptography · Auctions · Sealed-Bid · First-Price · Second-Price · Fairness · Blockchain

B. David—This work was supported by the Concordium Foundation and by the Independent Research Fund Denmark with grants number 9040-00399B (TrA^{2}C) and number 9131-00075B (PUMA).
L. Gentile—This work was supported by the Concordium Foundation.
M. Pourpouneh—This work was supported by the Center for Blockchains and Electronic Markets funded by the Carlsberg Foundation under grant no. CF18-1112.

© Springer Nature Switzerland AG 2022
G. Ateniese and D. Venturi (Eds.): ACNS 2022, LNCS 13269, pp. 727–747, 2022.
https://doi.org/10.1007/978-3-031-09234-3_36

1 Introduction

Auctions are a common way of allocating goods or services among a set of parties based on their bids, *e.g.*, bandwidth spectrum, antiques, paintings, and slots for advertisements in the context of web search engines or social networks [17]. In the simplest form, there is a single indivisible object, and each bidder has a *private valuation* for the object. One of the main desirable properties in designing an auction is *incentive compatibility*, that is the auction must be designed in a way that the participating parties can maximize their expected utilities by bidding their true valuations of the object. According to design, the auction can be categorized into open auctions, and sealed-bid auctions [31].

We focus on the case of sealed-bid auctions, constructing protocols where parties holding a private bid do not have to rely on trusted third parties to ensure bid privacy. In a sealed bid auction, each bidder communicates her bid to the auctioneer privately. Then, the auctioneer is expected to declare the highest bidder as the winner and not to disclose the losing bids. In particular, in the sealed-bid *first-price* auction, the bidder submitting the highest bid wins the auction and pays what she bids, while in the sealed-bid *second-price* auction (*i.e.*, the Vickrey auction [41]) the bidder submitting the highest bid wins the auction but pays the amount of the second-highest bid [30]. It is well-known that in the second-price auctions bidding truthfully is a dominant strategy, but no dominant strategy exists in the case of first-price auctions. Moreover, while in both first-price and second-price auctions, a dishonest auctioneer may disclose the losing bids, the second-price auction, in particular, highly depends on trusting the auctioneer. Indeed, a dishonest auctioneer may substitute the second-highest bid with a bid that is slightly smaller than the first bid to increase her revenue. Therefore, it may not be possible or expensive to apply it in certain scenarios. As a result, constructing cryptographic protocols for auctioneer-free and transparent auction solutions is of great interest.

1.1 Our Contributions

In this paper, we propose Fair Auctions via Secret Transactions (FAST), in which there is no trusted auctioneer and where rational adversaries are always incentivized to complete protocol execution through a *secret* collateral deposit. The proposed protocol is such that each party can make sure the winning bid is the actual bid submitted by the winning party, and malicious parties can be identified, financially punished and removed from the execution (guaranteeing a winner is always determined). Our contributions are summarized as follows:

- We propose using *secret* collateral deposits dependent on private bids inputs to ensure that the optimal strategy is for parties to complete the protocol.
- (Sect. 3) We propose methods for implementing a financial punishment mechanism based on secret deposits and standard public smart contracts, which can be used to ensure the fair execution of our protocols.

– (Sects. 4) We propose cheater identifiable and publicly verifiable sealed bid auction protocols compatible with our secret deposit approach and more efficient than the state-of-the-art [3]. Our protocols are guaranteed to terminate, finding the winner, and paying the seller even if cheating occurs.

To achieve fairness in an auction setting, we require each party to provide a *secret* deposit of an amount of cryptocurrency equal to the party's private bid plus the cost of executing the protocol. In case a party is found to be cheating, a smart contract automatically redistributes cheaters' deposits among the honest parties, the cheater is eliminated and the remaining parties re-execute the protocol using their initial bids/deposits. Having a bid-dependent deposit guarantees that it is always more profitable to execute the protocol honestly than to cheat (as analyzed in Sect. 5).

However, previous works that considered the use of cryptocurrency deposits for achieving fairness (e.g. [2,6,8,9,18,29]) crucially rely on deposits being public, thus using the same approach would reveal information about the bid. To overcome this, we propose using secret deposits that keep the value of the deposit secret until cheating is detected. Moreover, this ensures that the parties have sufficient funds to bid for the object (*e.g.*, in a second-price auction, a party could bid very high just to figure out what is the second-highest price is and then claim her submitted bid was just a mistake). Our protocols are publicly verifiable, *i.e.* it is possible to prove to the smart contract (and to any third party verifier) that a party has cheated.

In relation to previous works (discussed in Sect. 1.3), we emphasize that:

– While using deposits to achieve fairness represents a well-known technique, previous works considered public deposits only.
– Public deposits are not suitable for applications such as sealed-bid auctions since in order to achieve fairness, bid-dependent deposits are required, and public deposits would reveal information about the bid. For this reason, we introduce secret deposits, which represent a novel technique.
– From a sealed bid auction perspective, our protocol improves the state-of-the-art both in terms of efficiency and security guarantees, i.e., it achieves fairness (while in previous works the adversary may learn the outcome of the auction before honest parties and abort without suffering any consequences).
– No previous work in this setting considers adaptive adversaries since it would drastically increase the complexity of the protocol. For this reason, we focus on the static adversary case only.

1.2 Our Techniques

We start with a first-price sealed-bid auction protocol that builds on a simple passively secure protocol similar to that of SEAL [3] and compile it to achieve active security. However, we not only obtain an actively secure protocol but also add cheater identification and public verifiability properties. We use these properties to add our financial punishment mechanism with secret deposits to this protocol. Even though our protocol achieves stronger security guarantees

than SEAL (*i.e.*, sequential composability and fairness guarantees), it is more efficient than the SEAL protocol as shown in Sect. 6.

A Toy Example: Our protocol uses a modified version of the *Anonymous Veto Protocol* from [25] as a building block. The anonymous veto protocol allows a set of n parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$ to anonymously indicate whether they want to veto or not on a particular subject by essentially securely computing the logical-OR function of their inputs. In this protocol, each party $\mathcal{P}_i$ has an input bit $d_i \in \{0, 1\}$ with 0 indicating no veto and 1 indicating veto, and they wish to compute $\bigvee_{i=1}^{n} d_i$.

As proposed in [3], this simple anonymous veto protocol can be used for auctions by having parties evaluate their bids bit-by-bit, starting from the most significant bit and proceeding to execute the veto protocol for each bit in the following way: 1. Until there is no veto, all parties only veto (input $d_i = 1$ in the veto protocol) if and only if the current bit of their bid is 1; 2. After the first veto, a party only vetoes if the bit of her bid in the last time a veto happened was 1 *and* the current bit is also 1. In other words, in this toy protocol, parties stop vetoing once they realize that there is another party with a higher bid (*i.e.*, there was a veto in a round when their own bit were 0) and the party with the highest bid continues vetoing according to her bid until the last bit. Therefore, the veto protocol output represents the highest bid. However, a malicious party can choose not to follow the protocol, altering the output.

Achieving Active Security with Cheater Identification and Public Verifiability: To achieve active security with cheater identification and public verifiability, we depart from a simple passively secure protocol and compile it into an active secure protocol using NIZKs following an approach similar to that of [26,29]. This ensures that at every round of the protocol all parties' inputs are computed according to the protocol rules, including previous rounds' inputs and outputs. However, since the generic techniques from [26,29] yield highly inefficient protocols, we carefully construct tailor-made efficient non-interactive zero-knowledge proofs for our specific protocol, ensuring it to be efficient.

Incentivizing Correct Behaviour with Secret Deposits: In order to create incentives for parties to behave honestly, a deposit based on their bids is required. However, a public deposit would leak information about the parties' bids, which have to be kept secret. Hence, we do secret deposits as discussed below and keep the amount secret unless a party is identified as a cheater, in which case the cheater's deposit is distributed among the honest parties. The cheater is then eliminated and the protocol is re-executed with the remaining parties using their initial bids/deposits so that a winner is determined. This makes it rational not to cheat both in the case of first and second-price auctions, *i.e.*, cheating always implies a lower utility than behaving honestly (see Sect. 5).

Achieving On-Chain Efficiency: In order to minimize the amount of on-chain communication, an approach based on techniques from [5] is adopted. Every time a message is sent from a given party to the other parties, all of them sign the message received and send the signature to each other. Communication is only done on-chain (through the smart contract) in case of suspected cheating.

Secret Deposits to Public Smart Contracts: Since we use secret deposits based on confidential transactions [35], we need a mechanism to reveal the value of cheating parties' deposits to the smart contract so it can punish cheaters. We do that by secret sharing trapdoor information used to reveal this value using a publicly verifiable secret sharing (PVSS) scheme [15] that allows us to prove in zero-knowledge both that the shares are valid and that they contain the trapdoor for a given deposit. These shares are held by a committee that does not act unless cheating is detected, in which case the committee members are reimbursed for reconstructing the trapdoor with funds from the cheater's deposit itself. We discuss this approach in Sect. 3. Providing alternative methods for holding these deposits is an important open problem.

1.3 Related Work

Research on secure auctions started by the work of Nurmi and Salomaa [38] and Franklin and Reiter [23] in the late 1900s. However, in these first constructions, the auctioneers open all bids at the end of the protocol, which reveals the losing bids to all parties. Since then, many sealed bid auction protocols have been proposed to protect the privacy of the losing bids, *e.g.*, [1,4,27,32,33]. However, in most of these protocols, privacy is obtained by distributing the computation of the final outcome to a group of auctioneers.

A lot of work has been done to remove the role of the trusted parties, *e.g.*, by Brandt [11]. In these protocols, the bidders must compute the winning bid in a joint effort through emulating the role of the auctioneer. Moreover, the seller plays a role in the auction and it is assumed that the seller has no incentive to collude with other bidding parties. However, later by Dreier *et al.* [22] it was pointed out that if the seller and a group of bidding parties collude with each other, then they can learn the bids of other parties. Besides weak security guarantees, the main drawback of the protocol proposed by Brandt [11] is that it has exponential computational and communication complexities.

There have been implementations of auctions including [10], which have been deployed in practice for the annual sugar beets auction in Denmark. Other works [36] have considered the use of rational cryptography in enhancing privacy. Finally, the current state-of-the-art in protocols for secure First-Price Sealed-Bid Auctions was achieved in SEAL [3], which we compare with our protocols in detail in Sect. 6. To the best of our knowledge, none of these works considers incentives for the parties to complete the protocol or punishment for cheaters.

An often desired feature of Secure Multiparty Computation (MPC) is that if a cheating party obtains the output, then all the honest parties should do so as well. Protocols that guarantee this are also called *fair* and are known to be impossible to achieve with dishonest majorities [16]. Recently, Andrychowicz et al. [2] (and independently Bentov & Kumaresan [8]) initiated a line of research that aims at incentivizing fairness in MPC by imposing cryptocurrency-based financial penalties on misbehaving parties. A line of work [9,18] culminating in [6] improved the performance of this approach with respect to the amount of on-chain storage and size of the collateral deposits from each party, while others

obtained stronger notions of fairness [29]. However, all of these works focus on using *public* collateral deposits for incentivizing fairness, which is not possible for our application. Moreover, they rely on general-purpose MPC, while we provide a highly optimized specific purpose protocol for auctions with financial incentives. The protocols of [21,24] are also based on cryptocurrencies. The work of [24] is the closest to ours as it leverages a cryptocurrency to ensure fairness, but it relies on SGX trusted execution enclaves.

2 Preliminaries

Let $y \xleftarrow{\$} F(x)$ denote running the randomized algorithm F with input x and implicit randomness, obtaining the output y. When the randomness r is specified, we use $y \leftarrow F(x;r)$. For a set $\mathcal{X}$, let $x \xleftarrow{\$} \mathcal{X}$ denote x chosen uniformly at random from $\mathcal{X}$; and for a distribution $\mathcal{Y}$, let $y \xleftarrow{\$} \mathcal{Y}$ denote y sampled according to the distribution $\mathcal{Y}$. We denote concatenation of two values x and y by $x|y$. We denote negligible functions as $\mathrm{negl}(x)$. We denote two *computationally indistinguishable* ensembles $X = \{X_{\kappa,z}\}_{\kappa\in\mathbb{N},z\in\{0,1\}^*}$ and $Y = \{Y_{\kappa,z}\}_{\kappa\in\mathbb{N},z\in\{0,1\}^*}$ of binary random variables by $X \approx_c Y$. For a field $\mathbb{F}$ we denote by $\mathbb{F}[X]_{\leq m}$ the vector space of polynomials in $\mathbb{F}[X]$ of degree at most m.

2.1 Security Model and Setup Assumptions

We prove our protocol secure in the real/ideal simulation paradigm with sequential composition. This paradigm is commonly used to analyse cryptographic protocol security and provides strong security guarantees, namely that several instances of the protocol can be executed in sequence while preserving their security. To prove security, a real world and an ideal world are defined and compared. In the real world, the protocol π is executed with the parties, some of which are corrupted and controlled by the adversary $\mathcal{A}$. In the ideal world, the protocol is replaced by an ideal functionality $\mathcal{F}$ and a simulator $\mathcal{S}$ interacts with it. The ideal functionality $\mathcal{F}$ describes the behaviour that is expected from the protocol and acts as a trusted entity. A protocol π is said to securely realize the ideal functionality $\mathcal{F}$, if for every polynomial-time adversary $\mathcal{A}$ in the real world, there is a polynomial-time simulator $\mathcal{S}$ for the ideal world, such that the two worlds cannot be distinguished. In more detail, no probabilistic polynomial-time distinguisher $\mathcal{D}$ can have a non-negligible advantage in distinguishing the concatenation of the output of the honest parties and of the adversary $\mathcal{A}$ in the real world from the concatenation of the output of the honest parties (which come directly from $\mathcal{F}$) and of the simulator $\mathcal{S}$ in the ideal world. More details about this model are in [14]. Our protocol uses the Random Oracle Model (ROM). Note that adopting the UC model, as an alternative, requires to use UC-secure NIZK (instead of those described subsequently), but reduces the efficiency of the protocol. Also, previous works consider the sequential composability model only.

Adversarial Model: We consider *malicious* adversaries that may deviate from the protocol in any arbitrary way. Moreover, we consider the *static* case, where the adversary is only allowed to corrupt parties before protocol execution starts and parties remain corrupted (or not) throughout the execution. Moreover, we assume that parties have access to synchronous communication channels, *i.e.*, all messages are delivered within a given round with a known maximum delay.

Decisional Diffie Hellman (DDH) Assumption: The DDH problem consists in deciding whether $c = ab$ or $c \xleftarrow{\$} \mathbb{Z}_p$ in a tuple (g, g^a, g^b, g^c) where g is a generator of a group $\mathbb{G}$ of order p, and $a, b \xleftarrow{\$} \mathbb{Z}_p$. The DDH assumption states that the DDH problem is hard for every *PPT* distinguisher. It is well known that the DDH assumption implies the Discrete Logarithm assumption.

2.2 Building Blocks

Pedersen Commitments: Let p and q be large primes such that q divides $p-1$ and let $\mathbb{G}$ be the unique subgroup of $\mathbb{Z}_p^*$ of order q. All the computations in $\mathbb{G}$ are operations modulo p, however we omit the mod p to simplify the notation. Let g, h denote random generators of $\mathbb{G}$ such that nobody knows the discrete logarithm of h base g, *i.e.*, a value w such that $g^w = h$. The Pedersen commitment scheme [40] to an $s \in \mathbb{Z}_q$ is obtained by sampling $t \xleftarrow{\$} \mathbb{Z}_q$ and computing $,(s,t) = g^s h^t$. Hence, the commitment $,(s,t)$ is a value uniformly distributed in $\mathbb{G}$ and opening the commitment requires to reveal the values of s and t. The Pedersen commitments are additively homomorphic, *i.e.*, starting from the commitment to $s_1 \in \mathbb{Z}_q$ and $s_2 \in \mathbb{Z}_q$, it is possible to compute a commitment to $s_1 + s_2 \in \mathbb{Z}_q$, *i.e.*, $,(s_1,t_1)\cdot,(s_2,t_2) =,(s_1+s_2, t_1+t_2)$.

Simplified UTXO Model: In order to focus on the novel aspects of our protocol, we represent cryptocurrency transactions under a simplified version of the Bitcoin UTXO model [37]. For the sake of simplicity, we only consider operations of the "Pay to Public Key" (P2PK) output type, which we later show how to realize while keeping the values of transactions private. The formal description of the adopted simplified UTXO model is discussed in the full version [20].

Confidential Transactions: In the case of confidential transactions [35] the input and output amounts are kept secret using Pedersen commitments. However, in order to achieve public verifiability, the transactions contain a zero-knowledge proof that the sum of the inputs is equal to the sum of the outputs, and that all the outputs are between $[0, 2^l - 1]$ (which can be computed with Bullet Proofs [12]). Note that the input set In in confidential transactions can also be public, (*i.e.* $\mathsf{In} = \{(\mathtt{id}_1, \mathtt{in}_1), \ldots, (\mathtt{id}_m, \mathtt{in}_m)\}$), as long as the outputs are kept private. In particular, confidential transactions can be formally defined by modifying the simplified UTXO model described above as follows:

- **Representing inputs and outputs:** Set In is defined as $\mathsf{In} = \{(\mathtt{id}_1, \mathrm{com}(\mathtt{in}_1, r_{\mathtt{in}_1})), \ldots, (\mathtt{id}_m, \mathrm{com}(\mathtt{in}_m, r_{\mathtt{in}_m}))\}$ and set Out is defined as $\mathsf{Out} = \{(\mathrm{com}(\mathtt{out}_1, r_{\mathtt{out}_1}), Addr_1), \ldots, (\mathrm{com}(\mathtt{out}_n, r_{\mathtt{out}_n}), Addr_n)\}$.

- **Generate Transaction with $\mathsf{In}, \mathsf{Out}$:** Compute $\frac{\prod_{j=1}^{n} \mathrm{com}(\mathtt{out}_j, r_{\mathtt{out}_j})}{\prod_{i=1}^{m} \mathrm{com}(\mathtt{in}_i, r_{\mathtt{in}_i})} = \mathrm{com}(0, \sum_{j=1}^{n} r_{\mathtt{out}_j} - \sum_{i=1}^{m} r_{\mathtt{in}_i})$ with $r_{\mathtt{in}_i}, r_{\mathtt{out}_j} \xleftarrow{\$} \mathbb{Z}_q$, include in the transaction the randomness $\sum_{j=1}^{n} r_{\mathtt{out}_j} - \sum_{i=1}^{m} r_{\mathtt{in}_i}$ and the range proofs π guaranteeing that $\mathtt{out}_1, \cdots, \mathtt{out}_n$ are between $[0, 2^l - 1]$. The resulting transaction is then represented by $\mathsf{tx} = (\mathtt{id}, \mathsf{In}, \mathsf{Out}, \mathsf{Sig}, \sum_{j=1}^{n} r_{\mathtt{out}_j} - \sum_{i=1}^{m} r_{\mathtt{in}_i}, \pi)$.
- **Validate a Transaction tx:** Compute $\frac{\prod_{j=1}^{n} \mathrm{com}(\mathtt{out}_j, r_{\mathtt{out}_j})}{\prod_{i=1}^{m} \mathrm{com}(\mathtt{in}_i, r_{\mathtt{in}_i})} = \mathrm{com}(s, t)$ and check if the obtained commitments is equal to $\mathrm{com}(0, \sum_{j=1}^{n} r_{\mathtt{out}_j} - \sum_{i=1}^{m} r_{\mathtt{in}_i})$, guaranteeing that $\sum_{i=1}^{m} \mathtt{in}_i = \sum_{j=1}^{n} \mathtt{out}_j$, then check the validity of the range proofs π.
- **Spend a transaction output Out:** Parse $\mathsf{Out} = (\mathrm{com}(\mathtt{out}_i, r_{\mathtt{out}_i}), Addr_i)$. To spend Out, the commitment $\mathrm{com}(\mathtt{out}_i, r_{\mathtt{out}_i}) = g^{\mathtt{out}_i} h^{r_{\mathtt{out}_i}}$ has to be opened by revealing $\mathtt{out}_i$ and $r_{\mathtt{out}_i}$. Values $\mathtt{out}_i$ and $r_{\mathtt{out}_i}$ are included in a regular UTXO transaction and they are described in the full version [20]. Later on, this UTXO transaction can be validated by checking that $\mathtt{out}_i, r_{\mathtt{out}_i}$ is a valid opening of $\mathrm{com}(\mathtt{out}_i, r_{\mathtt{out}_i})$ and following the steps of a regular UTXO transaction validation.
- **Spend a transaction output Out with a NIZKPoK of $r_{\mathtt{out}_i}$:** Alternatively, an output $\mathsf{Out} = (\mathrm{com}(\mathtt{out}_i, r_{\mathtt{out}_i}), Addr_i)$ for which only $\mathtt{out}_i$ and $\hat{h} = h^{r_{\mathtt{out}_i}}$ (but not $r_{\mathtt{out}_i}$) are known can be spent if a NIZK π' proving knowledge of $r_{\mathtt{out}_i}$ is also available. Notice that knowing $\mathtt{out}_i$ is sufficient for validating the regular UTXO transaction created using Out as an input. Moreover, it can be checked that $g^{\mathtt{out}_i} h^{r_{\mathtt{out}_i}} = \mathrm{com}(\mathtt{out}_i, r_{\mathtt{out}_i})$ given $\mathtt{out}_i$ and $\hat{h} = h^{r_{\mathtt{out}_i}}$, while the proof π' guarantees that $\hat{h} = h^{r_{\mathtt{out}_i}}$ is well formed.[1] Values $\mathtt{out}_i$, $h^{r_{\mathtt{out}_i}}$ and the proof π' are included in a regular UTXO transaction generated and they are described in the full version [20]. Later on, this UTXO transaction can be validated by checking that $g^{\mathtt{out}_i} h^{r_{\mathtt{out}_i}} = \mathrm{com}(\mathtt{out}_i, r_{\mathtt{out}_i})$, checking that π' is valid and following the steps of a regular UTXO transaction validation.

Publicly Verifiable Secret Sharing (PVSS): In our work, we use the PVSS protocol π_{PVSS} from [15]. A PVSS protocol allows for a dealer to distribute encrypted shares to a set of parties in such a way that only one specific party can decrypt a share but any third party verifier can check that all shares are valid. Later on, each party can decrypt its corresponding share to allow for reconstruction while showing to any third-party verifier that the decrypted share corresponds to one of the initial encrypted shares. A deposit committee $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ will execute this protocol verifying and decrypting shares provided as part of our secret deposit mechanism (further discussed in Sect. 3). Since the parties in $\mathcal{C}$ executing π_{PVSS} must have public keys registered as part of a setup phase, we capture this requirement in $\mathcal{F}_{\mathsf{SC}}$ as presented in Sect. 2.2.

[1] In fact, showing such a proof of knowledge π' of $r_{\mathtt{out}_i}$ together with $h^{r_{\mathtt{out}_i}}$ and $\mathtt{out}_i$ makes it easy to adapt reduction of the binding property of the Pedersen commitment scheme to the Discrete Logarithm assumption. Instead of obtaining $r_{\mathtt{out}_i}$ from the adversary, the reduction simply extracts it from π'.

NIZK for PVSS Share Consistency CC**:** As part of our secret deposit mechanism (further discussed in Sect. 3), we will use a NIZK showing that shares computed with the PVSS protocol π_{PVSS} from [15] encode secrets g^m and h^r that are terms of a Pedersen commitment $c = g^m h^r$. Formally, given generators $g_1, \ldots, g_n, g, h$ of a cyclic group $\mathbb{G}_q$ of prime order q, pairwise distinct elements $\alpha_1, \ldots, \alpha_n$ in $\mathbb{Z}_q$ and a Pedersen commitment $c = g^m h^r$ known by prover and verifier, for $p(x)$ and m, r known by the prover, this NIZK is used to prove that $(\hat{\sigma}_1, \ldots, \hat{\sigma}_n) \in \left\{ \left(g_1^{p(\alpha_1)}, \ldots, g_n^{p(\alpha_n)} \right) : p \in \mathbb{Z}_q[X], p(-1) = g^m, p(-2) = h^r \right\}$. We denote this NIZK by $CC((g_i)_{i\in[n]}, (\alpha_i)_{i\in[n]}, g, h, c, (\hat{\sigma}_i)_{i\in[n]})$. Notice that this NIZK can be constructed using the techniques from [13] and integrated with the NIZK $LDEI$ (Low-Degree Exponent Interpolation) defined in π_{PVSS} [15].

Modelling a Stateful Smart Contract: We employ a stateful smart contract functionality $\mathcal{F}_{\mathsf{SC}}$ similar to that of [18] in order to model the smart contract that implements the financial punishment mechanism for our protocol. For the sake of simplicity, we assume that each instance of $\mathcal{F}_{\mathsf{SC}}$ is already parameterized by the address of the auctioneer party who will receive the payment for the auctioned good, as well as by the identities (and public keys) of the parties in a secret deposit committee $\mathcal{C}$ that will help the smart contract to open secret deposits given by parties in case cheating is detected. We also assume that $\mathcal{F}_{\mathsf{SC}}$ has a protocol verification mechanism pv for verifying the validity of protocol messages. For description of the $\mathcal{F}_{\mathsf{SC}}$ see the full version [20].

3 Secret Deposits in Public Smart Contracts

When using secret deposits as in our application, it is implied that there exists a secret trapdoor that can be used to reveal the value of such deposits (and transfer them). However, since we base our financial punishment mechanism on a standard public smart contract, we cannot expose the trapdoor to the smart contract. Instead, we propose that a committee $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ with $m/2 + 2$ honest members[2] holds this trapdoor in a secret shared form. This committee does not act unless a cheating party needs to be punished and the trapdoor needs to be reconstructed to allow the smart contract to transfer her collateral deposit. In this case, the committee can be reimbursed from the collateral funds. We present a practical construction following this approach. Proposing methods for keeping custody of such secret deposits is left as an important open problem.

A Possible Solution: A feasible but not practical approach to do this would be storing the trapdoor with the mechanism proposed in [7], where a secret is kept by obliviously and randomly chosen committees by means of a proactive secret sharing scheme where each current committee "encrypts the secret to the future" in such a way that the next committee can open it. However, it is also necessary

[2] We need $m/2+2$ honest members to instantiate our packed publicly verifiable secret sharing based solution where two group elements are secret shared with a single share vector.

to ensure that the secrets actually correspond to the trapdoor for the parties' deposits. Providing such proofs with the scheme of [7] would require expensive generic zero-knowledge techniques (or a trusted setup for a zk-SNARK).

Protocol $\Pi_{\mathcal{C}}$

Let $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ be the deposit committee members and $pk_{\mathcal{C}_1}, \ldots, pk_{\mathcal{C}_m}$ and $sk_{\mathcal{C}_1}, \ldots, sk_{\mathcal{C}_m}$ be their public keys and private keys, used to run π_{PVSS}, respectively. Moreover, let $pk'_{\mathcal{C}_1}, \ldots, pk'_{\mathcal{C}_m}$ and $sk'_{\mathcal{C}_1}, \ldots, sk'_{\mathcal{C}_m}$ be their public keys and private keys, used for signatures, respectively. The following steps are executed by $\mathcal{C}_j \in \mathcal{C}$:

- **Setup verification:** Upon receiving $(\text{SETUP}, sid, \mathcal{P}_i, \mathsf{tx}_i, pk_i, (\hat{\sigma}_{i1}, \ldots, \hat{\sigma}_{im}), LDEI_i, CC_i)$ from $\mathcal{P}_i, \mathcal{C}_j$ checks that tx_i is valid, verifies the shares $(\hat{\sigma}_{i1}, \ldots, \hat{\sigma}_{im})$ correctness with respect to the committee public keys $pk_{\mathcal{C}_1}, \ldots, pk_{\mathcal{C}_m}$ using the verification procedure of π_{PVSS} through $LDEI_i$ and verifies NIZK CC_i. If all the checks pass, compute the hashes $SH1_i = \mathcal{H}(\mathsf{tx}_i, pk_i)$ and $SH2_i = \mathcal{H}((\hat{\sigma}_{i1}, \ldots, \hat{\sigma}_{im}), LDEI_i, CC_i)$ and the signature $Sig_{\mathcal{C}_j,i} = sig_{sk'_{\mathcal{C}_j}}(SH1_i | SH2_i)$, then send $(\text{SETUP-VERIFICATION}, sid, Sig_{\mathcal{C}_j,i})$ to $\mathcal{P}_i$.
- **Share decryption:** Upon receiving $(\text{OPEN}, sid, \mathcal{P}_i)$ from $\mathcal{F}_{\mathsf{SC}}$, $\mathcal{C}_j$ uses the share decryption procedure from π_{PVSS} on $\hat{\sigma}_{ij}$, obtaining $\tilde{\sigma}_{ij}, DLEQ_{ij}$. and sending $(\text{SHARE-DECRYPTION}, sid, (\hat{\sigma}_{i1}, \ldots, \hat{\sigma}_{im}), LDEI_i, CC_i, \tilde{\sigma}_{ij}, DLEQ_{ij}))$ to $\mathcal{F}_{\mathsf{SC}}$.

Fig. 1. Protocol $\Pi_{\mathcal{C}}$

A Protocol Based on PVSS: As an alternative, we propose leveraging the structure of our confidential transaction based deposits to secret share their openings with a recent efficient publicly verifiable secret sharing (PVSS) scheme called Albatross [15]. Notice that the secret amount information b_i in these deposits is represented as a Pedersen commitment $g^{b_i}h^{r_i}$ and that the Albatross PVSS scheme also allows for sharing a group element g^s, while proving in zero-knowledge discrete logarithm relations involving g^s in such a way that they can be verified by any third party with access to the public *encrypted* share. Hence, we propose limiting the bid b_i bit length in such a way that we can employ the same trick as in *lifted ElGamal* and have each party $\mathcal{P}_i$ share both g^{b_i} and h^{r_i} with the Albatross PVSS while proving that their public encrypted shares correspond to a secret deposit $g^{b_i}h^{r_i}$. The validity of this claim can be verified by the committee $\mathcal{C}$ itself or the smart contract during Stage 1 - Setup. Later on, if b_i needs to be recovered, $\mathcal{C}$ can reconstruct g^{b_i}, brute force b_i (because it has a restricted bit-length) and deliver it to the smart contract while proving it has been correctly computed from the encrypted shares. As we explain in Sect. 2, recovering b_i and g^{r_i} along with the proofs of share validity is sufficient for transferring the secret deposit.

In Fig. 1, we present Protocol $\Pi_{\mathcal{C}}$ followed by the committee $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ and executed as part of Protocols Π_{FPA} described in Sect. 4. The interaction of the other parties $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ executing Protocols Π_{FPA} and Π_{SPA} with the committee $\mathcal{C}$ is described as part of Stage 1 - Setup of these protocols.

Selecting Committees: In order to focus on the novel aspects of our constructions, we assume that the smart contract captured by $\mathcal{F}_{\mathsf{SC}}$ described in the full version [20] is parameterized by a description of the committee $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ and the public keys corresponding to each committee member. Notice that in practice this committee can be selected by the smart contract from the set of parties executing the underlying blockchain consensus protocol. The problem of selecting committees in a permissionless blockchain scenario has been extensively addressed in both Proof-of-Stake *e.g.* [19,28] and Proof-of-Work [39] settings.

4 First-Price Auctions

In this section, we introduce our protocol for first-price auctions (while the case of second-price auctions is addressed in the full version [20]). We consider a setting with n parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$, where each party $\mathcal{P}_i$ has a l-bit bid $b_i = b_{i1}| \ldots |b_{il}$, where b_{ir} denotes the r-th bit of party $\mathcal{P}_i$'s bid.

Functionality $\mathcal{F}_{FPA}$

$\mathcal{F}_{FPA}$ operates with an auctioneer $\mathcal{P}_{AUC}$, a set of parties $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ who have bids $b_1, \ldots, b_n$ as input and where $b_i = b_{i1}| \ldots |b_{il}$ is the bit representation of b_i, as well as an adversary $\mathcal{S}_{FPA}$. $\mathcal{F}_{FPA}$ is parameterized by a bid bit-length l and keeps an initially empty list bids.

- **Setup (Bid Registration):** Upon receiving $(\text{BID}, sid, \mathsf{coins}(b_i + work))$ from $\mathcal{P}_i$ where $b_i \in \{0,1\}^l$ and $work$ is the amount required to compensate the cost of running the protocol for all the other parties, $\mathcal{F}_{FPA}$ appends b_i to bids.
- **First-Price Auction:** After receiving $(\text{BID}, sid, \mathsf{coins}(b_i + work))$ from all parties in $\mathcal{P}$, for $r = 1, \ldots, l$ $\mathcal{F}_{FPA}$ proceeds as follows:
  1. Select b_{wr}, *i.e.*, the r-th bit of the highest bid b_w in the list bids.
  2. Send $(\text{ROUND-WINNER}, sid, b_{wr})$ to all parties and $\mathcal{S}_{FPA}$.
  3. Check if $b_{wr} = 1$ and $b_{ir} = 0$ for $i = 1, \ldots, n \neq w$. If so, let $r_w = r$, that is the first position where b_w has a bit 1 and the second highest bid b_{w_2} has a bit 0, and send $(\text{LEAK-TO-WINNER}, sid, r_w)$ to $\mathcal{P}_w$.
  4. Send $(\text{ABORT?}, sid)$ to $\mathcal{S}_{FPA}$. If $\mathcal{S}_{FPA}$ answers with $(\text{ABORT}, sid, \mathcal{P}_i)$ where $\mathcal{P}_i$ is corrupted, remove b_i from bids, remove $\mathcal{P}_i$ from $\mathcal{P}$, send $(\text{ABORT}, sid, \mathcal{P}_i, \mathsf{coins}(\frac{b_i + work}{|\mathcal{P}|}))$ where $|\mathcal{P}|$ is the number of remaining parties to all other parties in $\mathcal{P}$, set again $r = 1$ and go to Step 1. If $\mathcal{S}_{FPA}$ answers with $(\text{PROCEED}, sid)$, if $r = l$ go to Payout, else increment r by 1 and go to Step 1.
- **Payout:** Send $(\text{REFUND}, sid, \mathcal{P}_w, \mathsf{coins}(b_i + work))$ to all parties $\mathcal{P}_i \neq \mathcal{P}_w$, send $(\text{REFUND}, sid, \mathsf{coins}(work))$ to $\mathcal{P}_w$, send $\mathsf{coins}(b_w)$ to $\mathcal{P}_{AUC}$, and halt.

Fig. 2. Functionality $\mathcal{F}_{FPA}$

Modelling Fair Auctions: First, we introduce an ideal model for fair auctions that we will use to prove the security of our protocol. For the sake of simplicity, when discussing this model, we use $\mathsf{coins}(n)$ to indicate n currency tokens being

transferred where n is represented in binary, instead of describing a full UTXO transaction. Our ideal functionality $\mathcal{F}_{FPA}$ is described in Fig. 2. This functionality models the fact that the adversary may choose to abort but all it may learn is that it was the winner and the most significant bit where its bid differs from the second-highest bid. Regardless of adversarial actions, an auction result is always obtained and the auctioneer (*i.e.*, the party selling the asset) is always paid. The second-price case is presented in the full version [20].

The Protocol: In Figs. 3, 4, 5 and 6, we construct a Protocol Π_{FPA} that realizes $\mathcal{F}_{FPA}$. This protocol is executed by n parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$, where each party $\mathcal{P}_i$ has a l-bit bid $b_i = b_{i1}| \ldots |b_{il}$ and a deposit committee $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ that helps open secret deposits from corrupted parties in the Recovery Stage. The protocol consists of 4 main stages plus a recovery stage, which is only executed in case of suspected (or detected) cheating. In the first stage, every party i sends to the smart contract a secret deposit, whose structure will be explained in detail later. In the second and third stages, all parties jointly compute the maximum bid (bit-by-bit) by using an anonymous veto protocol that computes a logical OR on private inputs. To this aim, the parties start from the most significant bit position. Then, they apply the anonymous veto protocol according to their bits b_{ir}, with 0 representing a no veto and 1 representing a veto. If the outcome of the veto protocol (*i.e.*, the logical-OR of the the inputs) is 1, then each party $\mathcal{P}_i$ with input $b_{ir} = 0$ figures out that there is at least another party $\mathcal{P}_k$ whose bid b_k is higher than b_i and $\mathcal{P}_i$ discovers that she cannot win the auction. Therefore, from this point on, $\mathcal{P}_i$ stops vetoing, disregarding her actual bit b_{ir} in the next rounds. Otherwise, $\mathcal{P}_i$ is expected to keep vetoing or not according to her bit b_{ir}. Finally, in Stage 4 the winning party $\mathcal{P}_w$ executes the payment to the auctioneer (*i.e.*, the party selling the asset). Throughout all stages, the parties must provide proofs that they have correctly computed all protocol messages. If a party is identified as dishonest at any point, the Recovery Stage has to be executed.

Security Analysis: It is clear that this protocol correctly computes the highest bid. The ideal smart contract enforces payment once a winner is determined and punishments otherwise. The security of this protocol is formally stated in the following theorem. A game-theoretical analysis is presented in Sect. 5, where it is shown that the best strategy for any rational party is to follow the protocol.

Theorem 1. *Under the DDH Assumption, Protocol Π_{FPA} securely computes $\mathcal{F}_{FPA}$ in the $\mathcal{F}_{\mathsf{SC}}$-hybrid, random oracle model against a malicious static adversary $\mathcal{A}$ corrupting all but one parties $\mathcal{P}_i \in \mathcal{P}$ and $m/2 - 2$ parties $\mathcal{C}_i \in \mathcal{C}$.*

Due to space limits, we leave the full proof to the full version [20].

5 Rational Strategies

In this section, we consider the incentives of parties in our protocols. Note that the set of bidders is fixed through the execution, *i.e.*, once the execution has started, even if it is required to re-execute the protocol, no new bid can be

Protocol Π_{FPA} (Off-chain messages exchange)

Protocol Π_{FPA} is executed with n parties $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$, where each party $\mathcal{P}_i$ has a l-bit bid $b_i = b_{i1}| \ldots |b_{il}$ and a deposit committee $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$. Parties $\mathcal{P}, \mathcal{C}$ interact among themselves and with a smart contract $\mathcal{F}_{\mathsf{SC}}$.

Off-chain messages exchange: To minimize the communication with the smart contract, an approach based on [5] is adopted. Let r be a generic round of the protocol, then each party $\mathcal{P}_i$ actually proceeds as follows when sending her messages:

- **Round**$_r$: each $\mathcal{P}_i$ sends $msg_{r,i}$, $sig_{sk_i}(msg_{r,i})$ to all the other parties;
- **Round**$_{r+1}$: all the other parties $\mathcal{P}_k$ for $k \in \{1, \ldots, n\} \setminus i$ sign the message received from party i and send $msg_{r,i}$, $sig_{sk_k}(msg_{r,i})$ to all the other parties, allowing them to check if party i sent no conflicting messages. Then, each party repeats from the instructions described in the previous round;
- **Conflicting messages**: in case $\mathcal{P}_i$ sends conflicting messages $msg_{r,i} \neq msg'_{r,i}$ to parties $\mathcal{P}_k \neq \mathcal{P}_{k'}$, $\mathcal{P}_k$ or $\mathcal{P}_{k'}$ send to the smart contract $msg_{r,i}$, $sig_{sk_i}(msg_{r,i})$ and $msg'_{r,i}$, $sig_{sk_i}(msg'_{r,i})$ as a proof that $\mathcal{P}_i$ was dishonest;
- **Evidence of a message**: in case it has to be proven that a message $msg_{r,i}$ has been sent by party $\mathcal{P}_i$ in round r, the other parties send to the smart contract the signatures $sig_{sk_1}(msg_{r,i}), \ldots, sig_{sk_n}(msg_{r,i})$ along with the message $msg_{r,i}$.

Fig. 3. Protocol Π_{FPA} (Off-chain messages exchange)

submitted, and it is therefore not possible to gain from the leaked information. Moreover, in case there is a cheating party, the protocols refund the honest parties with her deposit.

We now consider the utility of each party from participating in the protocol. The utility function of a generic party $\mathcal{P}_i$ in the first-price auction is $u_i^{FPA}(b_1, \ldots, b_n) = v_i - b_i$ if $b_i > \max_{j \neq i} b_j$ and 0 otherwise, while in the second-price auction is instead $u_i^{SPA}(b_1, \ldots, b_n) = v_i - \max_{j \neq i} b_j$ if $b_i > \max_{j \neq i} b_j$ and 0 otherwise, where v_i represents the $\mathcal{P}_i$'s private valuation of what is at stake in the auction. It is known that in the first-price auctions the optimal strategy for each rational party depends on their beliefs regarding other party's valuations, while in the second-price auction the optimal strategy for each party is to bid an amount equal to her valuation regardless of the strategy of other parties [31,34], *i.e.*, $b_i = v_i$.

In case a party $\mathcal{P}_i$ is honest, she always gets her deposit $work$ back. Then, if she is the winner, she gets what is at stake in the auction and pays b_i, while if she is not the winner, she gets her entire deposit $b_i + work$ back. Therefore, by following the protocol each rational party has a non-negative utility, *i.e.*, $u_i(b_1, \ldots, b_n) \geq 0$. However, if a party cheats her deposit $b_i + work$ is distributed among honest parties. Therefore, the utility of a cheating party, regardless of whether her bid is the highest or not, is $u_i(b_1, \ldots, b_n) = -(b_i + work) < 0$, which is strictly negative. Therefore, cheating is a dominated strategy for each party, *i.e.*, regardless of what other players do it always results in a lower utility.

Protocol Π_{FPA} (Stage 1)

Stage 1 - Setup: Deposit committee parties $\mathcal{C}_k \in \mathcal{C}$ first execute the Setup Verification step of $\Pi_{\mathcal{C}}$ from Figure 1. All parties $\mathcal{P}_i$ proceed as follows:

1. $\mathcal{P}_i$ sends a secret deposit containing their bid b_i, change *change* and a fee *work* to the smart contract through a confidential transaction (as described in Section 2.2). Let $Addr_i$ be the address associated to party i and $Addr_s$ be the address associated to the smart contract, $\mathcal{P}_i$ proceeds as follows:
   (a) $\mathcal{P}_i$ sends (PARAM, *sid*) to $\mathcal{F}_{\mathsf{SC}}$, receiving (PARAM, $sid, g, h, pk_{\mathcal{C}_1}, \ldots, pk_{\mathcal{C}_m}$).
   (b) $\mathcal{P}_i$ computes the bit commitments as $c_{ir} = g^{b_{ir}} h^{r_{ir}}$, with $r_{ir} \xleftarrow{\$} \mathbb{Z}_q$, to each bit b_{ir} of b_i, and the bid commitment as $c_i = \prod_{r=1}^{l} c_{ir}^{2^{l-r}} = g^{b_i} h^{\sum_{r=1}^{l} 2^{l-r} r_{ir}}$. Let r_{b_i} be equal to $\sum_{r=1}^{l} 2^{l-r} r_{ir}$. Then, c_i can be rewritten as $c_i = g^{b_i} h^{r_{b_i}} = \mathrm{com}(b_i, r_{b_i})$.
   (c) Define sets $\mathsf{In} = \{(\mathtt{id}_i, \mathtt{in}_i)\}$ and $\mathsf{Out} = \{(c_i, Addr_s), (work, Addr_s), (\mathrm{com}(change_i, r_{change_i}), Addr_i)\}$, where $c_i = \mathrm{com}(b_i, r_{b_i})$ is the commitment to the bid b_i previously computed at Step 1, *work* is the amount required to compensate the cost of running the protocol for all the other parties in $\mathcal{P}$ and in $\mathcal{C}$, $change = \mathtt{in}_i - b_i - work$ and $r_{change} \xleftarrow{\$} \mathbb{Z}_q$. Note that, in this case case, $\mathtt{in}_i$ and *work* are public, while b_i and *change* are private.
   (d) Compute $r_{out} = r_{b_i} + r_{change_i}$, so as to allow the other parties later to verify that the sum of the inputs is equal to the sum of the outputs, *i.e.* $c_i \cdot \mathrm{com}(change, r_{change}) \stackrel{?}{=} \mathrm{com}(\mathtt{in}_i - work, r_{out})$.
   (e) Compute proofs $(\pi_{b_i}, \pi_{change})$ showing that $b_i, change \in [0, 2^l - 1]$, set $\mathsf{tx}_i = (\mathtt{id}, \mathsf{In}, \mathsf{Out}, \mathsf{Sig}, r_{b_i} + r_{change_i}, \pi)$.
   (f) Compute the shares $(\hat{\sigma}_{i1}, \ldots, \hat{\sigma}_{im}, LDEI_i)$ of g^{b_i} and $h^{r_{b_i}}$ using the distribution procedure from π_{PVSS} with $pk_{\mathcal{C}_1}, \ldots, pk_{\mathcal{C}_m}$ received in step (a).
   (g) Compute $CC_i \leftarrow CC((pk_{\mathcal{C}_j})_{j\in[m]}, (j)_{j\in[m]}, g, h, c_i, (\hat{\sigma}_j)_{j\in[m]})$ to prove consistency among the shares $(\hat{\sigma}_{i1}, \ldots, \hat{\sigma}_{im})$ and the commitment terms g^{b_i} and $h^{r_{b_i}}$ from $c_i = g^{b_i} h^{r_{b_i}}$.
   (h) Send (SETUP, $sid, \mathcal{P}_i, \mathsf{tx}_i, pk_i, (\hat{\sigma}_{i1}, \ldots, \hat{\sigma}_{im}), LDEI_i, CC_i$) to each $\mathcal{C}_j \in \mathcal{C}$.
   (i) Upon receiving (SETUP-VERIFICATION, $sid, Sig_{\mathcal{C}_j,i}$) from all $\mathcal{C}_j \in \mathcal{C}$, compute $SH1_i = \mathcal{H}(\mathsf{tx}_i, pk_i)$ and $SH2_i = \mathcal{H}((\hat{\sigma}_{i1}, \ldots, \hat{\sigma}_{im}), LDEI_i, CC_i)$ and send (SETUP, $sid, \mathcal{P}_i, \mathsf{tx}_i, pk_i, SH1_i, SH2_i, Sig_{\mathcal{C}_1,i}, \ldots, Sig_{\mathcal{C}_m,i}$)) to $\mathcal{F}_{\mathsf{SC}}$. If a party $\mathcal{C}_a \in \mathcal{C}$ does not send this message, proceed to the Recovery Stage.
2. $\mathcal{P}_i$ samples $x_{ir} \xleftarrow{\$} \mathbb{Z}_q$ and computes $X_{ir} = g^{x_{ir}}$ for $r = 1, \ldots, l$, sending $c_{i1}, \cdots, c_{il}$, $X_{i1}, \cdots, X_{il}$ to all other parties.
3. Upon receiving all messages $c_{j1}, \cdots, c_{jl}$, $X_{j1}, \cdots, X_{jl}$ from other parties $\mathcal{P}_j$, $\mathcal{P}_i$ computes $Y_{jk} = \prod_{m=1}^{j-1} X_{mk} / \prod_{m=j+1}^{n} X_{mk}$ for $j = 1, \ldots, n, k = 1, \ldots, l$, and verifies for each other party $\mathcal{P}_j$ that $c_j = \prod_{k=1}^{l} c_{jk}^{2^{l-k}}$ for $j \in \{1, \ldots, n\} \setminus i$. If this verification fails or a message is not received, proceed to the Recovery Stage.

Fig. 4. Protocol Π_{FPA} (Stage 1)

The above analysis shows that it is not rational for an adversary $\mathcal{A}$ controlling a single party to deviate from the protocol. Next, we show that it is also the case for an adversary $\mathcal{A}$ controlling more than one party. Let $\mathcal{P}_i, \mathcal{P}_j$ be two parties controlled by $\mathcal{A}$ and let $v_{\mathcal{A}}$ be the valuation of the adversary for what is at stake

Protocol Π_{FPA} (Stages 2 and 3)

Stage 2 - Before First Veto: All parties $\mathcal{P}_i$, starting from the most significant bit b_{i1} and moving bit-by-bit to the least significant bit b_{il} of their bid $b_i = b_{i1}|\dots|b_{il}$, run in each round r the anonymous veto protocol until the outcome is a veto (*i.e.*, $V_r \neq 1$) for the first time. Therefore each party $\mathcal{P}_i$ proceeds as follows:

1. Compute v_{ir} as follows: if $b_{ir} = 0$ then $v_{ir} = Y_{ir}^{x_{ir}}$; if $b_{ir} = 1$ then $v_{ir} = g^{\bar{r}_{ir}}$ where $\bar{r}_{ir} \xleftarrow{\$} \mathbb{Z}_q$. Then generate NIZK proving that v_{ir} has been correctly computed $BV_{ir} \leftarrow BV\{b_{ir}, r_{ir}, x_{ir}, \bar{r}_{ir} \mid (\frac{c_{ir}}{g^{b_{ir}}} = c_{ir} = h^{r_{ir}} \wedge v_{ir} = Y_{ir}^{x_{ir}} \wedge X_{ir} = g^{x_{ir}}) \vee (\frac{c_{ir}}{g^{b_{ir}}} = \frac{c_{ir}}{g} = h^{r_{ir}} \wedge v_{ir} = g^{\bar{r}_{ir}})\}$, sending a message (v_{ir}, BV_{ir}) to all parties.
2. Upon receiving all messages (v_{kr}, BV_{kr}) from other parties $\mathcal{P}_k$, $\mathcal{P}_i$ checks the proofs BV_{kr} for $k \in \{1, \dots, n\} \backslash i$ and, if all checks pass, computes $V_r = \prod_{k=1}^{n} v_{kr}$ and then goes to Stage 3 if $V_r \neq 1$ (at least one veto), otherwise follows the steps in Stage 2 again until the round $r = l$. Note that, unless all the bids are equal to 0, at some point the condition $V_r \neq 1$ is satisfied. If a message is not received from party $\mathcal{P}_k$ or if BV_{kr} is invalid, proceed to the Recovery Stage.

Stage 3 - After First Veto: Let $\widehat{r}$ denote the last round at which there was a veto (*i.e.*, $V_{\widehat{r}} \neq 1$). All parties $\mathcal{P}_i$, starting from $b_{i\widehat{r}+1}$ and moving bit-by-bit to the least significant bit b_{il} of their bid $b_i = b_{i1}|\dots|b_{il}$, run in each round $r > \widehat{r}$ the anonymous veto protocol taking into account both the input bit b_{ir} and the declared input bit d_{ir}, defined as the value that satisfies the logical condition $(b_{ir} = 0 \wedge d_{ir} = 0) \vee (b_{ir} = 1 \wedge d_{i\widehat{r}} = 1 \wedge d_{ir} = 1) \vee (b_{ir} = 1 \wedge d_{i\widehat{r}} = 0 \wedge d_{ir} = 0)$, *i.e.*, each party $\mathcal{P}_i$ vetoes at round r iff she also vetoed at round $\widehat{r}$ (*i.e.*, $d_{i\widehat{r}} = 1$), and her current input bit $b_{ir} = 1$. Therefore, each $\mathcal{P}_i$ proceeds as follows:

1. Compute v_{ir} as follows: if $b_{ir} = 0$, then $v_{ir} = Y_{ir}^{x_{ir}}$; if $d_{i\widehat{r}} = 1 \wedge b_{ir} = 1$, then $v_{ir} = g^{\bar{r}_{ir}}$ where $\bar{r}_{ir} \xleftarrow{\$} \mathbb{Z}_q$; if $d_{i\widehat{r}} = 0 \wedge b_{ir} = 1$, then $v_{ir} = Y_{ir}^{x_{ir}}$. Then generate NIZK proving that v_{ir} has been correctly computed $AV_{ir} \leftarrow AV\{b_{ir}, r_{ir}, x_{ir}, \bar{r}_{i\widehat{r}}, \bar{r}_{ir}, x_{i\widehat{r}} \mid (\frac{c_{ir}}{g^{b_{ir}}} = c_{ir} = h^{r_{ir}} \wedge v_{ir} = Y_{ir}^{x_{ir}} \wedge X_{ir} = g^{x_{ir}}) \vee (\frac{c_{ir}}{g^{b_{ir}}} = \frac{c_{ir}}{g} = h^{r_{ir}} \wedge d_{i\widehat{r}} = g^{\bar{r}_{i\widehat{r}}} \wedge v_{ir} = g^{\bar{r}_{ir}}) \vee (\frac{c_{ir}}{g^{b_{ir}}} = \frac{c_{ir}}{g} = h^{r_{ir}} \wedge\ d_{i\widehat{r}} = Y_{i\widehat{r}}^{x_{i\widehat{r}}} \wedge X_{i\widehat{r}} = g^{x_{i\widehat{r}}} \wedge v_{ir} = Y_{ir}^{x_{ir}} \wedge X_{ir} = g^{x_{ir}})\}$, sending a message (v_{ir}, AV_{ir}) to all parties.
2. Upon receiving all messages (v_{kr}, AV_{kr}) from other parties $\mathcal{P}_k$, $\mathcal{P}_i$ checks the proofs AV_{kr} for $k \in \{1, \dots, n\} \backslash i$ and, if all checks pass, computes $V_r = \prod_{k=1}^{n} v_{kr}$, following the steps in Stage 3 again until round $r = l$. If a message is not received from party $\mathcal{P}_k$ or if AV_{kr} is invalid, proceed to the Recovery Stage.

Fig. 5. Protocol Π_{FPA} (Stages 2 and 3)

in the auction. Without loss of generality let $b_i > b_j$. If $\mathcal{A}$ does not deviate from the protocol, then her utility is either 0 (in case neither b_i nor b_j is the winning bid) or $v_{\mathcal{A}} - b_i$ (in case b_i is the winning bid). Instead, if $\mathcal{A}$ deviates from the protocol by making $\mathcal{P}_i$ dropout, in case b_j is not the second-highest bid, then her utility is $-(work + b_i)$. If b_j is the second-highest bid, $\mathcal{A}$ gets what is at stake in the auction but her utility is $v_{\mathcal{A}} - (b_i + work + b_j)$. Therefore $\mathcal{A}$ always prefers to behave honestly.

Protocol Π_{FPA} (Stages 4 and Recovery)

Stage 4 - Output: At this point, each party $\mathcal{P}_i$ knows the value of V_r for each round $r = 1, \cdots, l$ and the protocol proceeds as follows:

1. $\mathcal{P}_i$ computes the winning bid as $b_w = b_{w1}| \cdots |b_{wl}$, such that $b_{wr} = 1$ if $V_r \neq 1$ and $b_{wr} = 0$ if $V_r = 1$, and sends b_w to all other parties (causing all parties $\mathcal{P}_k$ to sign b_w and send $sig_{sk_k}(b_w)$ to each other). We denote by $\mathcal{P}_w$ the winning party (*i.e.* the party whose bid is b_w).
2. $\mathcal{P}_w$ opens the commitment to her bid $\text{com}(b_w, r_{b_w})$ towards the smart contract by sending $(\text{OUTPUT}, sid, \mathcal{P}_w, b_w, r_{b_w}, \{sig_{sk_k}(b_w)\}_{k\in[n]})$ to $\mathcal{F}_{\mathsf{SC}}$.
3. If $\mathcal{P}_w$ does not open her commitment or if multiple parties open their commitments, $\mathcal{P}_i$ proceeds to the Recovery Stage.
4. Finally, all parties who honestly completed the execution of the protocol receive a refund of their deposit from the smart contract, apart from the winning party, who only receives a refund equivalent to the *work* funds.

Recovery Stage: Parties $\mathcal{C}_i \in \mathcal{C}$ listen to $\mathcal{F}_{\mathsf{SC}}$ and execute the Share Decryption step of $\Pi_{\mathcal{C}}$ from Figure 1 if requested. In case a party $\mathcal{P}_i \in \mathcal{P}$ is suspected of cheating, the Recovery stage is executed as follows to identify the cheater depending on the exact suspected cheating:

- **Missing message or signatures**: a message msg_{ri} or a signature $sig_{sk_i}(msg_{r-1,i'})$, on a message $msg_{r-1,i'}$ by $\mathcal{P}'_i$, expected to be sent in round r by $\mathcal{P}_i$ is not received by $\mathcal{P}_k$. Then, $\mathcal{P}_k$ sends to $\mathcal{F}_{\mathsf{SC}}$ the message $(\text{RECOVERY-MISSING}, sid, msg, \{sig_{sk_k}(msg)\}_{k\in[n]})$, where msg is the last message signed by all parties and waits for $\mathcal{F}_{\mathsf{SC}}$ to request the missing message. In that way, $\mathcal{P}_i$ is expected to send msg_{ri} or $sig_{sk_i}(msg_{r-1,i'})$ to $\mathcal{F}_{\mathsf{SC}}$. If no action is taken, $\mathcal{P}_i$ is identified as a cheater.
- **Conflicting messages or Invalid message**: In round r, $\mathcal{P}_i$ sends conflicting messages msg_{ri}, $sig_{sk_i}(msg_{ri})$ and msg'_{ri}, $sig_{sk_i}(msg'_{ri})$ to different parties $\mathcal{P}_k$ and $\mathcal{P}'_k$. In this case, $\mathcal{P}_k$ and $\mathcal{P}'_k$ set the conflicting messages as a proof of cheating $\pi_c = (msg_{ri}, sig_{sk_i}(msg_{ri}), msg'_{ri}, sig_{sk_i}(msg'_{ri}))$. Otherwise, $\mathcal{P}_i$ sends an invalid message $msg_{ri}, sig_{sk_i}(msg_{ri})$ to $\mathcal{P}_k$ (*i.e.* the message does not follow the structure described in the protocol for messages in round r), $\mathcal{P}_k$ uses this message as a proof of cheating $\pi_c = (msg_{ri}, sig_{sk_i}(msg_{ri}))$. $\mathcal{P}_k$ sends $(\text{RECOVERY-CHEAT}, sid, \mathcal{P}_i, \pi_c)$ to the smart contract and $\mathcal{P}_i$ is identified as a cheater.

Every party $\mathcal{P}_i$ identified as a cheater loses her whole deposit $(b_i + work)$, which is distributed to the other parties by $\mathcal{F}_{\mathsf{SC}}$, and the protocol continues as follows:

- **Re-execution (unknown b_w)**: in case b_w has not been computed, the protocol is re-executed from Stage 2 excluding the parties identified as cheaters.
- **Complete payment (known b_w but unknown $\mathcal{P}_w$)**: in case b_w has been computed but $\mathcal{P}_w$ does not send $(\text{OUTPUT}, sid, \mathcal{P}_w, b_w, r_{b_w}, \{sig_{sk_k}(b_w)\}_{k\in[n]})$ to $\mathcal{F}_{\mathsf{SC}}$, all $\mathcal{P}_i \in \mathcal{P}$ compute a NIZK $NW_i \leftarrow NW\{x_{i1}, \ldots, x_{il} \mid (V_1 = 1 \wedge v_{i1} = Y_{i1}^{x_{i1}}) \vee \ldots \vee (V_l = 1 \wedge v_{i1} = Y_{i1}^{x_{i1}})\}$ showing that they are not the winner. Then they send to $\mathcal{F}_{\mathsf{SC}}$ $(\text{RECOVERY-PAYMENT}, sid, NW_i)$. The winner $\mathcal{P}_w$ (in case it is identified) or all parties $\mathcal{P}_i$ who do not act (in case $\mathcal{P}_w$ is not identified) are identified as dishonest and lose their deposits, which are distributed among the honest parties.

Fig. 6. Protocol Π_{FPA} (Stages 4 and Recovery)

Note that *it is necessary to have the deposit amount at least equal to the bid.* Indeed, let d be any deposit amount smaller than b_i. Then the utility of $\mathcal{A}$ by making $\mathcal{P}_i$ drop out the protocol is $v_{\mathcal{A}} - (d + work + b_j)$, while it is $v_{\mathcal{A}} - b_i$ by behaving honestly. Therefore, in case $d + work + b_j < b_i$, $\mathcal{A}$ prefers to deviate from the protocol to increase her utility. A similar argument shows that in the second-price auction, $\mathcal{A}$ always prefers to act honestly.

6 Complexity Analysis and Comparison to Other Protocols

In this section, we present concrete estimates for the computational and communication complexity of our first and second-price auction protocols, *i.e.*, Π_{FPA} and Π_{SPA}, respectively. We show that, in the first-price case, Π_{FPA} is more efficient than the state-of-the-art protocol SEAL [3]. In the second-price case, we show that Π_{SPA} only incurs a small overhead (dominated by re-executing one round) over Π_{FPA}. Note that the complexity of Stages 2 and 3 is based on the NIZK constructions available in the full version [20].

Table 1. First-price auction computational complexity comparison in terms of exponentiations performed by a party $\mathcal{P}_i \in \mathcal{P}$: n is the number of parties, l is the total number of rounds in Stages 2 and 3 (*i.e.*, bit-length of bids), τ is the number of rounds in Stage 2.

	Stage 1	**Stage 2**	**Stage 3**	**Total**
FAST	$nl + l + 8\log l + 2$	$\tau(8 + 10n)$	$(l - \tau)(19 + 22n)$	$23nl + 20l + 8\log l - 11\tau - 12n\tau + 2$
SEAL [3]	$11l + 12nl$	$\tau(17 + 20n)$	$(l - \tau)(33 + 36n)$	$48nl + 44l - 16\tau - 16n\tau$

Table 2. First-price auction communication complexity comparison in terms of transmitted bits by a party $\mathcal{P}_i \in \mathcal{P}$: n is the number of parties, l is the total number of rounds in Stages 2 and 3 (*i.e.*, the bit-length of bids), τ is the number of rounds of Stage 2, $|\mathbb{G}|$ and $|\mathbb{Z}_q|$ indicate the bit-length of elements $g \in \mathbb{G}$ and $z \in \mathbb{Z}_q$ respectively, κ id the security parameter, as defined in Sect. 2.

	Stage 1	**Stage 2**	**Stage 3**	**Total**
FAST	$n((2l + 10)\lvert\mathbb{G}\rvert + 3\kappa + 4\log l)$	$n\tau(\lvert\mathbb{G}\rvert + 6\lvert\mathbb{Z}_q\rvert)$	$n(l - \tau)(\lvert\mathbb{G}\rvert + 11\lvert\mathbb{Z}_q\rvert)$	$n\big(\lvert\mathbb{G}\rvert(3l + 10) + \lvert\mathbb{Z}_q\rvert(11l - 5\tau) + 3\kappa + 4\log l\big)$
SEAL [3]	$17nl\lvert\mathbb{G}\rvert$	$23n\tau\lvert\mathbb{G}\rvert$	$36n(l - \tau)\lvert\mathbb{G}\rvert$	$(53nl - 13n\tau)\lvert\mathbb{G}\rvert$

The First-Price Case: A concrete estimate of computational complexity is shown in Table 1 and one for communication complexity is shown in Table 2.

We estimate these concrete complexities in terms of the number of exponentiations performed by a party $\mathcal{P}_i$ and of the number of bits transmitted by a party $\mathcal{P}_i$ in an execution of protocol Π_{FPA}, respectively. Moreover, we compare the complexity of our protocol with SEAL [3], which is the current state-of-the-art protocol for first-price sealed-bid auctions. In a similar way to our protocol, SEAL requires all parties to jointly compute the maximum bid bit-by-bit and is subdivided into a Stage 1 devoted to the setup, a Stage 2 identifying the rounds of the protocol before the first veto and a Stage 3 identifying the rounds of the protocol after the first veto. Hence, we highlight the differences in terms of complexity stage by stage. Note that, in order to make the communication complexities of the two protocols comparable, both of them have been expressed in terms of $|\mathbb{G}|$. Finally, FAST has an additional Stage 4 guaranteeing that the payment from the winning party $\mathcal{P}_w$ to the auctioneer is executed. On the other hand, SEAL does not guarantee this property. In particular, Stage 4 requires 1 exponentiation per party and has a communication complexity equal to $2(n-1)|\mathbb{G}|$.

The Second-Price Case: The computational and communication complexities of the proposed second-price auction are still linear in the number of agents. That is, assuming that at round r, there is a party who is the only one that is vetoing, then the parties have to re-run the r^{th} round with one less party. More precisely, by following the notation of Table 1 and 2, let τ be the number of rounds in Stage 2, then the computational complexity of Stage 1 and Stage 2 is similar to the first-price auction, that is $nl+l+8\log l+2$ for Stage 1, and $8\tau+10n\tau$ for Stage 2. Let r, be the number of rounds until there is only a single party who is veto-ing. Therefore the computational complexity of Stage 3 is $19r+22nr$ until there is only a single veto. After this, the parties have to run the protocol with one less party, i.e., $n-1$ parties. Depending on the bid structure of the remaining $n-1$ parties, the protocol is either in Stage 2 or Stage 3. Let τ' denote the number of rounds until the remaining $n-1$ parties get a veto. Then the computational complexity for these τ' rounds would be $8\tau'+10(n-1)\tau'$, and for the remaining $l-(\tau+\tau'+r)$ it would be $19\big(l-(\tau+\tau'+r)\big)+22(n-1)\big(l-(\tau+\tau'+r)\big)$. Using the same notation, a similar argument follows for the communication complexity per party in the case of the second-price auction.

References

1. Abe, M., Suzuki, K.: M + 1-st price auction using homomorphic encryption. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 115–124. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45664-3_8
2. Andrychowicz, M., Dziembowski, S., Malinowski, D., Mazurek, L.: Secure multiparty computations on bitcoin. In: 2014 IEEE Symposium on Security and Privacy, pp. 443–458. IEEE Computer Society Press, May 2014
3. Bag, S., Hao, F., Shahandashti, S.F., Ray, I.G.: Seal: sealed-bid auction without auctioneers. IEEE Trans. Inf. Forensics Secur. **15**, 2042–2052 (2019)

4. Baudron, O., Stern, J.: Non-interactive private auctions. In: Syverson, P. (ed.) FC 2001. LNCS, vol. 2339, pp. 364–377. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46088-8_28
5. Baum, C., David, B., Dowsley, R.: A framework for universally composable publicly verifiable cryptographic protocols. IACR Cryptol. ePrint Arch. **2020**, 207 (2020)
6. Baum, C., David, B., Dowsley, R.: Insured MPC: efficient secure computation with financial penalties. In: Bonneau, J., Heninger, N. (eds.) FC 2020. LNCS, vol. 12059, pp. 404–420. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51280-4_22
7. Benhamouda, F., et al.: Can a public blockchain keep a secret? In: Pass, R., Pietrzak, K. (eds.) TCC 2020. LNCS, vol. 12550, pp. 260–290. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64375-1_10
8. Bentov, I., Kumaresan, R.: How to use bitcoin to design fair protocols. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 421–439. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44381-1_24
9. Bentov, I., Kumaresan, R., Miller, A.: Instantaneous decentralized poker. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10625, pp. 410–440. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70697-9_15
10. Bogetoft, P., Damgård, I., Jakobsen, T., Nielsen, K., Pagter, J., Toft, T.: A practical implementation of secure auctions based on multiparty integer computation. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 142–147. Springer, Heidelberg (2006). https://doi.org/10.1007/11889663_10
11. Brandt, F.: Fully private auctions in a constant number of rounds. In: Wright, R.N. (ed.) FC 2003. LNCS, vol. 2742, pp. 223–238. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45126-6_16
12. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy, pp. 315–334. IEEE Computer Society Press, May 2018
13. Camenisch, J., Stadler, M.: Proof systems for general statements about discrete logarithms. Technical Report/ETH Zurich, Department of Computer Science, vol. 260 (1997)
14. Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptol. **13**(1), 143–202 (2000)
15. Cascudo, I., David, B.: ALBATROSS: publicly AttestabLe BATched randomness based on secret sharing. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12493, pp. 311–341. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64840-4_11
16. Cleve, R.: Limits on the security of coin flips when half the processors are faulty (extended abstract). In: 18th ACM STOC, pp. 364–369. ACM Press, May 1986
17. Cramton, P., et al.: Spectrum auctions. In: Handbook of Telecommunications Economics, vol. 1, pp. 605–639 (2002)
18. David, B., Dowsley, R., Larangeira, M.: Kaleidoscope: an efficient poker protocol with payment distribution and penalty enforcement. In: Meiklejohn, S., Sako, K. (eds.) FC 2018. LNCS, vol. 10957, pp. 500–519. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-662-58387-6_27
19. David, B., Gaži, P., Kiayias, A., Russell, A.: Ouroboros praos: an adaptively-secure, semi-synchronous proof-of-stake blockchain. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10821, pp. 66–98. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_3
20. David, B., Gentile, L., Pourpouneh, M.: FAST: fair auctions via secret transactions. Cryptology ePrint Archive, Report 2021/264 (2021). https://ia.cr/2021/264

21. Deuber, D., Döttling, N., Magri, B., Malavolta, G., Thyagarajan, S.A.K.: Minting mechanism for proof of stake blockchains. In: Conti, M., Zhou, J., Casalicchio, E., Spognardi, A. (eds.) ACNS 2020. LNCS, vol. 12146, pp. 315–334. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-57808-4_16
22. Dreier, J., Dumas, J.-G., Lafourcade, P.: Brandt's fully private auction protocol revisited. J. Comput. Secur. **23**(5), 587–610 (2015)
23. Franklin, M.K., Reiter, M.K.: The design and implementation of a secure auction service. IEEE Trans. Softw. Eng. **22**(5), 302–312 (1996)
24. Galal, H.S., Youssef, A.M.: Trustee: full privacy preserving Vickrey auction on top of ethereum. In: Bracciali, A., Clark, J., Pintore, F., Rønne, P.B., Sala, M. (eds.) FC 2019. LNCS, vol. 11599, pp. 190–207. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-43725-1_14
25. Hao, F., Zieliński, P.: A 2-round anonymous veto protocol. In: Christianson, B., Crispo, B., Malcolm, J.A., Roe, M. (eds.) Security Protocols 2006. LNCS, vol. 5087, pp. 202–211. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04904-0_28
26. Ishai, Y., Ostrovsky, R., Zikas, V.: Secure multi-party computation with identifiable abort. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 369–386. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44381-1_21
27. Juels, A., Szydlo, M.: A two-server, sealed-bid auction protocol. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 72–86. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36504-4_6
28. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: a provably secure proof-of-stake blockchain protocol. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 357–388. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_12
29. Kiayias, A., Zhou, H.-S., Zikas, V.: Fair and robust multi-party computation using a global transaction ledger. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 705–734. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_25
30. Klemperer, P.: Auctions: Theory and Practice. Princeton University Press, Princeton (2004)
31. Krishna, V.: Auction Theory. Academic Press, Cambridge (2009)
32. Kurosawa, K., Ogata, W.: Bit-slice auction circuit. In: Gollmann, D., Karjoth, G., Waidner, M. (eds.) ESORICS 2002. LNCS, vol. 2502, pp. 24–38. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45853-0_2
33. Lipmaa, H., Asokan, N., Niemi, V.: Secure Vickrey auctions without threshold trust. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 87–101. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36504-4_7
34. Mas-Colell, A., Whinston, M.D., Green, J.R., et al.: Microeconomic Theory, vol. 1. Oxford University Press, New York (1995)
35. Maxwell, G.: Confidential transactions (2016). https://people.xiph.org/~greg/confidential_values.txt
36. Miltersen, P.B., Nielsen, J.B., Triandopoulos, N.: Privacy-enhancing auctions using rational cryptography. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 541–558. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_32
37. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
38. Nurmi, H., Salomaa, A.: Cryptographic protocols for Vickrey auctions. Group Decis. Negot. **2**(4), 363–373 (1993). https://doi.org/10.1007/BF01384489

39. Pass, R., Shi, E.: Hybrid consensus: efficient consensus in the permissionless model. In: Richa, A.W. (ed.) 31st International Symposium on Distributed Computing, DISC 2017, volume 91 of LIPIcs, Vienna, Austria, 16–20 October 2017, pp. 39:1–39:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017)
40. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_9
41. Vickrey, W.: Counterspeculation, auctions, and competitive sealed tenders. J. Finance **16**(1), 8–37 (1961)