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. Auctioning an asset with sealed bids has been shown to be economically optimal but requires trusting a auctioneer who analyzes the bids and determines the winner. Many privacy preserving computation protocols for auctions have been proposed, aiming at eliminating 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 guarantee that the winner is determines 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. Our constructions are more efficient than the state-of-the-art even though they achieve stronger security guarantees, *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.

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

⋆ 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).
⋆⋆ This work was supported by the Concordium Foundation.
⋆⋆⋆ This work was supported by the Center for Blockchains and Electronic Markets funded by the Carlsberg Foundation under grant no. CF18-1112.

advertisements in the context of web search engines or social networks [29]. 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 [46].

We focus on the case of sealed-bid auctions, constructing protocols where parties who have 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 bid, while in the sealed bid *second price* auction (*i.e.*, the Vickrey auction [62]) the bidder submitting the highest bid wins the object but pays what the second highest bid is [45]. 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, which may have a commercial value. 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.
- (Section 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 fair execution of our protocols.
- (Sections 4 and 5) We propose a cheater identifiable and publicly verifiable sealed-bid auction protocols compatible with our secret deposit approach and more efficient than the state-of-the-art [4]. Our protocols are guaranteed to terminate, finding the winner and paying the seller even if cheating happens.

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 a case 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 Appendix F). However, in previous works that considered the use of cryptocurrency deposits for achieving fairness (e.g. [11, 3, 48, 47, 49, 12, 9, 30, 7, 44, 27], parameterizing the deposit in such a way would reveal information about the bid, since these works crucially rely on deposits being public. 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 claims her submitted bid was just a mistake). Our protocols are constructed in such a way that it is possible to prove to the smart contract that a party has cheated.

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 [4] 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 Section 6. Later on, we extend this protocol to the second price case with a very low performance overhead.

A Toy Example: Our protocol uses a modified version of the *Anonymous Veto Protocol* from [39] 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 $\mathcal{P}_\mathbf{i}$ has an input bit $d_\mathbf{i} \in \{0, 1\}$ with 0 indicating no veto and 1 indicating veto, and they wish to compute $\bigvee_{\mathbf{i}=1}^{n} d_\mathbf{i}$. As proposed in [4], 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 $d_\mathbf{i} = 1$ in the veto protocol) if and only if the current bit of their bid is 1; 2. After the first veto happens, a party only vetoes if the bit of their 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 their 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: In order 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 [37, 41, 44]. 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 [37, 41, 44] yield highly inefficient protocols, we carefully construct tailor made efficient non-interactive zero knowledge proofs for our specific protocol, obtaining an efficient protocol.

Incentivizing Correct Behavior with Secret Deposits: In order to create incentive 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 Appendix F).

Achieving On-Chain Efficiency: In order to minimize the amount of on-chain communication, an approach based on techniques from [6] 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 [53], 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 [24] 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 Section 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 [56] and Franklin and Reiter [34] 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, 5, 40, 42, 51, 50]. However, in most of these protocols the privacy is obtained by distributing the computation of the final outcome to a group of auctioneers.

A lot of work has been done in order to remove the role of the trusted parties, including the proposed protocols by Brandt [19, 17, 18], Brandt and Sandholm [20] and Bag *et al.* [4]. 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.* [32] 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, a main drawback of the protocol proposed by Brandt [18] is that it has exponential computational and communication complexities. There has been practical implementations of auctions in practice including [15, 14], which have been deployed in practice for the annual sugar beets auction in Denmark. Other works [54] 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 [4], which we compare with our protocols in detail in Section 6. However, to the best of our knowledge, none of these works consider 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 which guarantee this are also called *fair* and are known to be impossible to achieve with dishonest majorities [28]. Recently, Andrychowicz et al. [3] (and independently Bentov & Kumaresan [11]) 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 [48, 47, 49, 12, 9, 30] culminating in [7] 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 [44, 27]. 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 closest work to ours is [35], in the sense that it also leverages cryptocurrencies to ensure fairness. However, the protocol of [35] relies on SGX trusted execution enclaves, which are known to be broken [61].

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 $x|y$. A function $f(x)$ is negligible in x if $f(x)$ is positive and for every positive polynomial $p(x) \in \text{poly}(x)$ there exists a $x' \in \mathbb{N}$ such that $\forall x \geq x' : \ f(x) < 1/p(x)$, we denote such functions as $\text{negl}(x)$. Two 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 are said to be *computationally indistinguishable*, denoted by $X \approx_c Y$, if for all z it holds that $|\ \Pr[\mathcal{D}(X_{\kappa,z}) = 1] - \Pr[\mathcal{D}(Y_{\kappa,z}) = 1]\ |$ is negligible in the security parameter κ for every non-uniform probabilistic polynomial-time (PPT) distinguisher $\mathcal{D}$. 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 instance of the protocol can be executed in sequence while preserving their security. In order 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 $\mathcal{A}$. On the other hand, 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 behavior 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 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 can be found in [23, 36]. Our protocol uses the Random Oracle Model (ROM) [8].

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.

Decisional Diffie Hellman (DDH) Assumption: The DDH problem consists in deciding whether $c = ab$ or $c \xleftarrow{\$} \mathbb{Z}$ in a tuple (g, g^a, g^b, g^c) where g is a

generator of a group $\mathbb{G}$ of order p, and $a, b \xleftarrow{\$} \mathbb{Z}$. 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 [58] to an $s \in \mathbb{Z}_q$ is obtained by sampling $t \xleftarrow{\$} \mathbb{Z}_q$ and computing $\mathrm{com}(s,t) = g^s h^t$. Hence, the commitment $\mathrm{com}(s,t)$ 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.*, $\mathrm{com}(s_1, t_2) \cdot \mathrm{com}(s_2, t_2) = \mathrm{com}(s_1 + s_2, t_1 + t_2)$.

Digital Signatures: Let k be a *security parameter* and $\mathcal{M} \subseteq \{0,1\}^+$ be a *message space*, *i.e.*, the set of messages to which the signature may be applied. A signature scheme *SIG* [38] is a tuple of three polynomial-time algorithms:

- A *key generation algorithm* $\mathrm{Gen}(\kappa)$ that outputs a pair sk and pk of matching *secret* and *public* keys;
- A *signature algorithm* $\mathrm{Sig}(sk, msg)$ that takes the secret key sk and the message msg and outputs the signature $\sigma = Sig_{sk}(msg)$;
- A *verification algorithm* $\mathrm{Ver}(pk, msg, \sigma)$ that returns $True$ if and only if $\sigma = Sig_{sk}(msg)$, *i.e.*, σ is a signature of message msg computed with the secret key paired with the public key pk.

A signature is *correct* if $\mathrm{Ver}(pk, msg, sig_{sk}(msg)) = True$ for any security parameter k, any pair sk and pk of matching secret and public keys and any message $msg \in \mathcal{M}$.

Non-interactive Zero Knowledge Proofs of Knowledge: Let $\mathcal{L}$ be a language in NP and $R_{\mathcal{L}}$ be a relationship such that $\mathcal{L} = \{x \mid \exists w : (x,w) \in R_{\mathcal{L}}\}$. A zero-knowledge proof of knowledge protocol $\Sigma = (P, V)$ for a language $\mathcal{L}$ allows a prover P to demonstrate to a verifier V that $x \in \mathcal{L}$ provided that the prover knows a witness w such that $(x,w) \in R_{\mathcal{L}}$ and with the following properties:

- Completeness: if $(x,w) \in R_{\mathcal{L}}$ then the proof generated by P is accepted by the verifier V with overwhelming probability;
- Soundness: it is computationally hard for the prover P to prove to the verifier V that $(x,w) \in R_{\mathcal{L}}$ when $(x,w) \notin R_{\mathcal{L}}$.
- Zero knowledge: there exists a *PPT* simulator S that, by using rewinding, takes as input x but not w and generates a proof that $(x,.) \in R_{\mathcal{L}}$ for some w, *i.e.*, the verifier V does not learn w but can check if $(x,.) \in R_{\mathcal{L}}$ for some w;

– Proof of knowledge: there exists a PPT simulator S that interacts with a copy of the prover P and extracts, by using rewinding, the witness w, *i.e.*, the verifier V can check if the prover P knows a witness w such that $(x, w) \in R_{\mathcal{L}}$;

In our work we follow the approach of Camenisch and Stadler [22] based on the Fiat-Shamir heuristc [33] in order to obtain non interactive zero knowledge (NIZK) proofs of knowledge for discrete logarithm relations. We will use these NIZKs in forcing parties to execute our protocols correctly using the GMW methodology [37]. Notice that parties must provide such NIZKs proving they have executed each round of the protocol correctly, and an invalid NIZK is also a publicly verifiable proof that the party has misbehaved. We will be using two relations for two main stages of the protocol, which we describe below.

NIZK for Stage 2 - Before First Veto: In Stage 2 of the protocol, we need a NIZK proving knowledge of either $b_{\mathbf{ir}}, r_{\mathbf{ir}}, x_{\mathbf{ir}}$ such that $\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = c_{\mathbf{ir}} = h^{r_{\mathbf{ir}}} \wedge v_{\mathbf{ir}} = Y_{\mathbf{ir}}^{x_{\mathbf{ir}}} \wedge X_{\mathbf{ir}} = g^{x_{\mathbf{ir}}}$ or of $b_{\mathbf{ir}}, r_{\mathbf{ir}}, \bar{r}_{\mathbf{ir}}$ such that $\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = \frac{c_{\mathbf{ir}}}{g} = h^{r_{\mathbf{ir}}} \wedge v_{\mathbf{ir}} = g^{\bar{r}_{\mathbf{ir}}}$, where $v_{\mathbf{ir}}, c_{\mathbf{ir}}, X_{\mathbf{ir}}, g, Y_{\mathbf{ir}}$ are public. We denote this NIZK by,

$$BV\{b_{\mathbf{ir}}, r_{\mathbf{ir}}, x_{\mathbf{ir}}, \bar{r}_{\mathbf{ir}} \,|(\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = c_{\mathbf{ir}} = h^{r_{\mathbf{ir}}} \wedge v_{\mathbf{ir}} = Y_{\mathbf{ir}}^{x_{\mathbf{ir}}} \wedge X_{\mathbf{ir}} = g^{x_{\mathbf{ir}}}) \vee (\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = \frac{c_{\mathbf{ir}}}{g} = h^{r_{\mathbf{ir}}} \wedge v_{\mathbf{ir}} = g^{\bar{r}_{\mathbf{ir}}})\}.$$

A detailed construction of this NIZK is discussed in Appendix A.1.

NIZK for Stage 3 - After First Veto: In Stage 3 of the protocol, we require a NIZK proving knowledge of either $b_{\mathbf{ir}}, r_{\mathbf{ir}}, x_{\mathbf{ir}}$ such that $\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = c_{\mathbf{ir}} = h^{r_{\mathbf{ir}}} \wedge v_{\mathbf{ir}} = Y_{\mathbf{ir}}^{x_{\mathbf{ir}}} \wedge X_{\mathbf{ir}} = g^{x_{\mathbf{ir}}}$, or of $b_{\mathbf{ir}}, r_{\mathbf{ir}}, \bar{r}_{\hat{\mathbf{ir}}}, \bar{r}_{\mathbf{ir}}$ such that $\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = \frac{c_{\mathbf{ir}}}{g} = h^{r_{\mathbf{ir}}} \wedge d_{\hat{\mathbf{ir}}} = g^{\bar{r}_{\hat{\mathbf{ir}}}} \wedge v_{\mathbf{ir}} = g^{\bar{r}_{\mathbf{ir}}}$, or of $b_{\mathbf{ir}}, r_{\mathbf{ir}}, x_{\hat{\mathbf{ir}}}, x_{\mathbf{ir}}$ such that $\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = \frac{c_{\mathbf{ir}}}{g} = h^{r_{\mathbf{ir}}} \wedge \ d_{\hat{\mathbf{ir}}} = Y_{\hat{\mathbf{ir}}}^{x_{\hat{\mathbf{ir}}}} \wedge X_{\hat{\mathbf{ir}}} = g^{x_{\hat{\mathbf{ir}}}} \wedge v_{\mathbf{ir}} = Y_{\mathbf{ir}}^{x_{\mathbf{ir}}} \wedge X_{\mathbf{ir}} = g^{x_{\mathbf{ir}}}$. We denote this NIZK by,

$$\begin{aligned} AV\{b_{\mathbf{ir}}, r_{\mathbf{ir}}, x_{\mathbf{ir}}, \bar{r}_{\hat{\mathbf{ir}}}, \bar{r}_{\mathbf{ir}}, x_{\hat{\mathbf{ir}}} \,|&(\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = c_{\mathbf{ir}} = h^{r_{\mathbf{ir}}} \wedge v_{\mathbf{ir}} = Y_{\mathbf{ir}}^{x_{\mathbf{ir}}} \wedge X_{\mathbf{ir}} = g^{x_{\mathbf{ir}}}) \vee \\ &(\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = \frac{c_{\mathbf{ir}}}{g} = h^{r_{\mathbf{ir}}} \wedge d_{\hat{\mathbf{ir}}} = g^{\bar{r}_{\hat{\mathbf{ir}}}} \wedge v_{\mathbf{ir}} = g^{\bar{r}_{\mathbf{ir}}}) \vee \\ &(\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = \frac{c_{\mathbf{ir}}}{g} = h^{r_{\mathbf{ir}}} \wedge \ d_{\hat{\mathbf{ir}}} = Y_{\hat{\mathbf{ir}}}^{x_{\hat{\mathbf{ir}}}} \wedge X_{\hat{\mathbf{ir}}} = g^{x_{\hat{\mathbf{ir}}}} \wedge \\ &v_{\mathbf{ir}} = Y_{\mathbf{ir}}^{x_{\mathbf{ir}}} \wedge X_{\mathbf{ir}} = g^{x_{\mathbf{ir}}})\}. \end{aligned}$$

A detailed construction of this NIZK is discussed in Appendix A.2.

NIZK for Recovery - Proof of Not Winning: In case the winning bid b_w is determined but the winner $\mathcal{P}_w$ does not reveal herself, the honest parties in our auction protocol will prove in zero knowledge that they are not the winner. In order to do so, they use a NIZK $NW_{\mathbf{i}} \leftarrow NW\{x_{\mathbf{i}1}, \ldots, x_{\mathbf{i}l} \mid (V_1 = 1 \wedge v_{\mathbf{i}1} =$

$Y_{\mathrm{i1}}^{x_{\mathrm{i1}}}) \vee \ldots \vee (V_l = 1 \wedge v_{\mathrm{i1}} = Y_{\mathrm{i1}}^{x_{\mathrm{i1}}})\}$ that can be constructed using the techniques from [22]. We do not describe this NIZK construction nor estimate its complexity because it is used only in a corner case of cheating where the honest parties are reimbursed for generating and verifying such proofs.

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 [55]. 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 Appendix B.

Confidential transactions: In the case of confidential transactions [53] 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 [21]). 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 $\mathsf{Out} = (\mathrm{com}(\mathtt{out}_i, r_{\mathtt{out}_i}), Addr_i)$. In order 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 generated as described in Appendix B. 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[3]. Values $\mathtt{out}_i$, $h^r_{\mathtt{out}_i}$ and the proof π' are included in a regular UTXO transaction generated as described in Appendix B. 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.

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.

Publicly Verifiable Secret Sharing (PVSS): In our work, we use the PVSS protocol π_{PVSS} from [24], which is described in detail in Appendix C. 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 correpsonding 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 Section 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 Section 2.3.

NIZK for PVSS share consistency *CC*: As part of our secret deposit mechanism (further discussed in Section 3), we will use a NIZK showing that shares computed with the PVSS protocol π_{PVSS} from [24] encode secrets g^m and h^r that are terms of a Pedersen commitment $c = g^m h^r$. Formally, this NIZK is used to prove that, 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, such 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 [22] and integrated with the NIZK $LDEI$ already used in π_{PVSS} (presented in Appendix C).

2.3 Modeling a Stateful Smart Contract

We employ a stateful smart contract functionality $\mathcal{F}_{\mathsf{SC}}$ similar to that of [30] in order to model the smart contract that implements the financial punishment

[3] 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'.

Functionality $\mathcal{F}_{\mathsf{SC}}$ (Stages 1,2,3 and 4)

$\mathcal{F}_{\mathsf{SC}}$ is executed with parties $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$, a deposit committee members $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ and parameterized by committee public keys $pk_{\mathcal{C}_1}, \ldots, pk_{\mathcal{C}_m}$ used to run π_{PVSS} and $pk'_{\mathcal{C}_1}, \ldots, pk'_{\mathcal{C}_m}$ used for signatures. $\mathcal{F}_{\mathsf{SC}}$ is also parameterized by its own address $Addr_s$, the address of the Auctioneer $Addr_{auc}$, timeout limit τ, a protocol verification algorithm pv that given a protocol transcript outputs 1 if it is valid and replies 0 otherwise, and values $work_{\mathcal{P}}$ and $work_{\mathcal{C}}$ for the costs of executing the protocol for the parties in $\mathcal{P}$ and $\mathcal{C}$, respectively. Once execution starts $\mathcal{F}_{\mathsf{SC}}$ samples random generators $g, h \xleftarrow{\$} \mathbb{G}$ for a group $\mathbb{G}$.

- **Stage 1 - Setup:**
  - Upon receiving (PARAM, sid) from $\mathcal{P}_{\mathbf{i}}$, $\mathcal{F}_{\mathsf{SC}}$ returns $(\text{PARAM}, sid, g, h, pk_{\mathcal{C}_1}, \ldots, pk_{\mathcal{C}_m})$.
  - Upon receiving the message $(\text{SETUP}, sid, \mathcal{P}_{\mathbf{i}}, \mathsf{tx}_{\mathbf{i}}, pk_{\mathbf{i}}, SH1_{\mathbf{i}}|SH2_{\mathbf{i}}, Sig_{\mathcal{C}_1,\mathbf{i}}, \ldots, Sig_{\mathcal{C}_m,\mathbf{i}})$ from $\mathcal{P}_{\mathbf{i}}$, $\mathcal{F}_{\mathsf{SC}}$ checks that this message is valid according to pv. If all checks pass, $\mathcal{F}_{\mathsf{SC}}$ continues the execution. Otherwise, it proceeds to the Recovery Stage otherwise. After receiving this message from all $\mathcal{P}_{\mathbf{i}} \in \mathcal{P}$, $\mathcal{F}_{\mathsf{SC}}$ proceeds to the next steps.
- **Stages 2 and 3:** During these Stages, $\mathcal{F}_{\mathsf{SC}}$ only expects messages in case of suspected cheating, eventually proceeding to the Recovery Stage.
- **Stage 4 - Output:**
  - First Price Auction: Upon receiving $(\text{OUTPUT}, sid, \mathcal{P}_w, b_w, r_{b_w}, \{sig_{sk_k}(b_w)\}_{k\in[n]})$ from $\mathcal{P}_w$, $\mathcal{F}_{\mathsf{SC}}$ verifies that b_w, r_{b_w} is a valid opening to the commitment c_w in tx_w and that all signatures $\{sig_{sk_k}(b_w)\}_{k\in[n]}$ are valid. If all checks pass, $\mathcal{F}_{\mathsf{SC}}$ performs a transaction towards the Auctioneer paying b_w tokens to $Addr_{auc}$, refunds $\mathcal{P}_w$ with $work_{\mathcal{P}}$ tokens and refunds all the other parties $\mathcal{P}_{\mathbf{i}} \in \mathcal{P} \setminus \mathcal{P}_w$ with their full $\mathsf{tx}_{\mathbf{i}}$ transaction. Otherwise, proceed to the Recovery Stage.
  - Second Price Auction: Upon receiving $(\text{OUTPUT}, sid, \mathcal{P}_w, \mathsf{tx}_{pay}, \{sig_{sk_k}(\mathcal{P}_w|b_{w_2})\}_{k\in[n]})$ from $\mathcal{P}_w$, $\mathcal{F}_{\mathsf{SC}}$ verifies that that the range proof π_{pay} is valid and that $\text{com}(b_w, r_{b_w})/\text{com}(change'_w, r_{change'_w}) = \text{com}(b_{w_2}, r_{b_w} - r_{change'_w})$. If the checks pass, then $\mathcal{F}_{\mathsf{SC}}$ refunds all the other parties $\mathcal{P}_{\mathbf{i}} \in \mathcal{P} \setminus \mathcal{P}_w$ with their full $\mathsf{tx}_{\mathbf{i}}$ transaction and performs transaction tx_w paying b_{w_2} to the Auctioneer and refunding $\mathcal{P}_w$ with $work_{\mathcal{P}}$ tokens. If the check does not pass, then proceed to the Recovery Stage.

Fig. 1: Functionality $\mathcal{F}_{\mathsf{SC}}$ (Stages 1,2,3 and 4).

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. $\mathcal{F}_{\mathsf{SC}}$ is described in Figure 1.

Functionality $\mathcal{F}_{\mathsf{SC}}$ (Recovery)

– **Recovery:** Upon receiving one of the following messages $\mathcal{F}_{\mathsf{SC}}$ from a party $\mathcal{P}_i \in \mathcal{P}$ acts as described:
  - (RECOVERY-MISSING, $sid, msg, \{sig_{sk_k}(msg)\}_{k\in[n]}$): $\mathcal{F}_{\mathsf{SC}}$ sends (REQUEST, sid) to each $\mathcal{P}_{\mathbf{i}} \in \mathcal{P}$. If $\mathcal{P}_{\mathbf{i}}$ does not send the missing message $msg_{\mathbf{ri}}$ or $sig_{sk_{\mathbf{i}}}(msg_{\mathbf{r}-1,\mathbf{i}'})$ before a timeout τ or sends an invalid message according to pv, it is considered a cheater.
  - (RECOVERY-CHEAT, $sid, \mathcal{P}_j, \pi_c$): in case $\mathcal{P}_j$ has been accused to cheat by sending conflicting messages $msg_{\mathbf{r}j} \neq msg'_{\mathbf{r}j}$ then $\pi_c = (msg_{\mathbf{r}j}, sig_{sk_j}(msg_{\mathbf{r}j}), msg'_{\mathbf{r}j}, sig_{sk_j}(msg'_{\mathbf{r}j}))$, while in case $\mathcal{P}_j$ has been accused to cheat by sending an invalid message $msg_{\mathbf{r}j}$ according to pv then $\pi_c = (msg_{\mathbf{r}j}, sig_{sk_j}(msg_{\mathbf{r}j}))$. If π_c is a valid proof of cheating, then $\mathcal{P}_j$ is identified as a cheater, otherwise the sender $\mathcal{P}_i$ identified as a cheater.
  - (RECOVERY-PAYMENT, $sid, NW_{\mathbf{i}}$): $\mathcal{F}_{\mathsf{SC}}$ verifies that the proof of not winning $NW_{\mathbf{i}}$ is valid, *i.e.*, the sender $\mathcal{P}_i$ is not the winner. After a timeout τ counted from the moment the first message is received, all parties $\mathcal{P}_k$ who did not send a valid message (RECOVERY-PAYMENT, $sid, NW_{\mathbf{i}}$) are considered cheaters (*i.e.* either the corrupted $\mathcal{P}_w$ in case all other parties sent a valid $NW_{\mathbf{i}}$ or all parties who did not collaborate).
  - (RECOVERY-DISHONEST-WINNER, $sid, NW_{\mathbf{i}}$): *(second price case only)* $\mathcal{F}_{\mathsf{SC}}$ verifies that the proof of not winning $NW_{\mathbf{i}}$ is valid, *i.e.*, the sender $\mathcal{P}_i$ is not the winner. Then, $\mathcal{F}_{\mathsf{SC}}$ all parties $\mathcal{P}_k$ who did not send a valid message (RECOVERY-DISHONEST-WINNER, $sid, NW_{\mathbf{i}}$) are considered cheaters.

  If a party $\mathcal{P}_{\mathbf{i}}$ is a cheater, $\mathcal{F}_{\mathsf{SC}}$ collaborates with $\mathcal{C}$ to open $\mathcal{P}_{\mathbf{i}}$'s deposit $\mathsf{tx}_{\mathbf{i}}$ by sending (OPEN, $sid, \mathcal{P}_{\mathbf{i}}$) to each $\mathcal{C}_j \in \mathcal{C}$, takes its full deposit, reimburses each $\mathcal{C}_i \in \mathcal{C}$ with $work_{\mathcal{C}}$ tokens and reimburses each party $\mathcal{P} \setminus \mathcal{P}_{\mathbf{i}}$ with an equal share of the remaining tokens. In particular, $\mathcal{F}_{\mathsf{SC}}$ proceeds as follows:
  - Upon receiving (SHARE-DECRYPTION, $sid, (\hat{\sigma}_{\mathbf{i}1}, \ldots, \hat{\sigma}_{\mathbf{i}m}), LDEI_{\mathbf{i}}, CC_{\mathbf{i}}, \tilde{\sigma}_{\mathbf{i}j}, DLEQ_{\mathbf{i}j}$) from $\mathcal{C}_j$, $\mathcal{F}_{\mathsf{SC}}$ verifies that $SH2_{\mathbf{i}} = \mathcal{H}((\hat{\sigma}_{\mathbf{i}1}, \ldots, \hat{\sigma}_{\mathbf{i}m}), LDEI_{\mathbf{i}}, CC_{\mathbf{i}})$ where $SH2_{\mathbf{i}}$ is the one sent from $\mathcal{P}_{\mathbf{i}}$ in Stage 1 - Setup.
  - Use the share decryption verification procedure from π_{PVSS} to identify $2+n/2$ valid shares $\tilde{\sigma}_{\mathbf{i}j}$ (by verifying $DLEQ_{\mathbf{i}j}$) and then uses the secret reconstruction algorithm from π_{PVSS} to reconstruct $g^{b_{\mathbf{i}}}$ and $g^{r_{b_{\mathbf{i}}}}$. Next, $\mathcal{F}_{\mathsf{SC}}$ recovers $b_{\mathbf{i}}$ from $g^{b_{\mathbf{i}}}$ (since the length l of the bid is limited).
  - The deposit of the cheating party $\mathcal{P}_{\mathbf{i}}$ is distributed among the honest parties $\mathcal{P} \setminus \mathcal{P}_{\mathbf{i}}$ and $\mathcal{C}$ by spending the confidential transaction output $(c_{\mathbf{i}}, Addr_s)$ of $\mathsf{tx}_{\mathbf{i}}$. Indeed, in order to spend $(c_{\mathbf{i}}, Addr_s)$ it is sufficient to reveal $b_{\mathbf{i}}$, $h^{r_{b_{\mathbf{i}}}}$ and providing $(\hat{\sigma}_{\mathbf{i}1}, \ldots, \hat{\sigma}_{\mathbf{i}m}), LDEI_{\mathbf{i}}, CC_{\mathbf{i}}, \tilde{\sigma}_{\mathbf{i}j}, DLEQ_{\mathbf{i}j}$ to prove that $h^{r_{b_{\mathbf{i}}}}$ contains the same $r_{b_{\mathbf{i}}}$ of the initial commitment $c_{\mathbf{i}}$.

  $\mathcal{F}_{\mathsf{SC}}$ now expects the execution to restart with parties $\mathcal{P} \setminus \mathcal{P}_{\mathbf{i}}$ only.

Fig. 2: Functionality $\mathcal{F}_{\mathsf{SC}}$ (Recovery).

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[4] 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 in order to allow the smart contract to transfer their collateral deposit. In this case, the committee itself can be reimbursed from the collateral funds. We present a practical construction following this approach. Proposing more 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 [10], 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 to ensure that the secrets actually corresponds to the trapdoor for the parties' deposits. Providing such proofs with the scheme of [10] 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 $(\text{SETUP}, sid, \mathcal{P}_\mathbf{i}, \mathsf{tx}_\mathbf{i}, pk_\mathbf{i}, (\hat{\sigma}_{\mathbf{i}1}, \ldots, \hat{\sigma}_{\mathbf{i}m}), LDEI_\mathbf{i}, CC_\mathbf{i})$ from $\mathcal{P}_\mathbf{i}$, $\mathcal{C}_j$ checks that $\mathsf{tx}_\mathbf{i}$ is valid, verifies the shares $(\hat{\sigma}_{\mathbf{i}1}, \ldots, \hat{\sigma}_{\mathbf{i}m})$ 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 $LDEI_\mathbf{i}$ and verifies NIZK $CC_\mathbf{i}$. If all the checks pass, compute the hashes $SH1_\mathbf{i} = \mathcal{H}(\mathsf{tx}_\mathbf{i}, pk_\mathbf{i})$ and $SH2_\mathbf{i} = \mathcal{H}((\hat{\sigma}_{\mathbf{i}1}, \ldots, \hat{\sigma}_{\mathbf{i}m}), LDEI_\mathbf{i}, CC_\mathbf{i})$ and the signature $Sig_{\mathcal{C}_j,\mathbf{i}} = sig_{sk'_{\mathcal{C}_j}}(SH1_\mathbf{i}|SH2_\mathbf{i})$, then send $(\text{SETUP-VERIFICATION}, sid, Sig_{\mathcal{C}_j,\mathbf{i}})$ to $\mathcal{P}_\mathbf{i}$.
- **Share decryption:** Upon receiving $(\text{OPEN}, sid, \mathcal{P}_\mathbf{i})$ from $\mathcal{F}_{\mathsf{SC}}$, $\mathcal{C}_j$ uses the share decryption procedure from π_{PVSS} on $\hat{\sigma}_{\mathbf{i}j}$, obtaining $\tilde{\sigma}_{\mathbf{i}j}, DLEQ_{\mathbf{i}j}$. and sending $(\text{SHARE-DECRYPTION}, sid, (\hat{\sigma}_{\mathbf{i}1}, \ldots, \hat{\sigma}_{\mathbf{i}m}), LDEI_\mathbf{i}, CC_\mathbf{i}, \tilde{\sigma}_{\mathbf{i}j}, DLEQ_{\mathbf{i}j}))$ to $\mathcal{F}_{\mathsf{SC}}$.

Fig. 3: 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

[4] 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.

openings with a recent efficient publicly verifiable secret sharing (PVSS) scheme called Albatross [24]. Notice that the secret amount information $b_{\mathbf{i}}$ in these deposits is represented as a Pedersen commitment $g^{b_{\mathbf{i}}}h^{r_{\mathbf{i}}}$ 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 $b_{\mathbf{i}}$ bit length in such a way that we can employ the same trick as in *lifted ElGamal* and have each party $\mathcal{P}_{\mathbf{i}}$ share both $g^{b_{\mathbf{i}}}$ and $h^{r_{\mathbf{i}}}$ with the Albatross PVSS while proving that their public encrypted shares correspond to a secret deposit $g^{b_{\mathbf{i}}}h^{r_{\mathbf{i}}}$. 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 $b_{\mathbf{i}}$ needs to be recovered, $\mathcal{C}$ can reconstruct $g^{b_{\mathbf{i}}}$, brute force $b_{\mathbf{i}}$ (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 Section 2, recovering $b_{\mathbf{i}}$ and $r_{\mathbf{i}}$ along with the proofs of share validity is sufficient for transferring the secret deposit.

In Figure 3, 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} (resp. Π_{SPA}) described in Section 4 (resp. Section 5). 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 Section 2.3 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 [43, 31, 26] and Proof-of-Work [57] settings.

4 Main Protocol for First Price Auctions

In this section, we introduce our main protocol focusing on the case of first price auctions (while the case of Second Price auctions is addressed in Section 5). We consider a setting with n parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$, where each party $\mathcal{P}_{\mathbf{i}}$ has a l-bit bid $b_{\mathbf{i}} = b_{\mathbf{i}1}| \ldots |b_{\mathbf{i}l}$, where $b_{\mathbf{ir}}$ denotes the $\mathbf{r}$th bit of party $\mathcal{P}_{\mathbf{i}}$'s bid.

Modelling Fair Auctions: First, we introduce an ideal model for fair auctions that we will use to prove 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 Figure 4. 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

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

Fig. 4: Functionality $\mathcal{F}_{FPA}$.

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 Section 5.

The Protocol: In Figures 5, 6, 7 and 8, 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 $\mathcal{P}_\mathbf{i}$ has a l-bit bid $b_\mathbf{i} = b_{\mathbf{i}1}| \ldots |b_{\mathbf{i}l}$ 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 $\mathbf{i}$ sends to the smart contract a secret deposit, whose structure will be explained in details later. In the second and third stage, 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 $b_{\mathbf{ir}}$, 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 $\mathcal{P}_\mathbf{i}$ with input $b_{\mathbf{ir}} = 0$ figures out that there is at least another party $\mathcal{P}_k$ whose bid b_k is higher than $b_\mathbf{i}$ and $\mathcal{P}_\mathbf{i}$ discovers that she cannot win the auction. Therefore, from this point on, $\mathcal{P}_\mathbf{i}$ stops vetoing, disregarding her actual bit $b_{\mathbf{ir}}$ in the next rounds. Otherwise, $\mathcal{P}_\mathbf{i}$ is expected to keep vetoing or not according to her bit $b_{\mathbf{ir}}$.

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 (using the NIZKs described in Section 2.2). 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, which is proven in Appendix D. A game theoretical analysis is presented in Appendix F, 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 $\mathcal{C}_i \in \mathcal{C}$.*

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

Protocol Π_{FPA} is executed n parties $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$, where each party $\mathcal{P}_\mathbf{i}$ has a l-bit bid $b_\mathbf{i} = b_{\mathbf{i}1}| \ldots |b_{\mathbf{i}l}$ 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: In order to minimize the amount of communication with the smart contract, an approach based on techniques from [6] is adopted. Every time we write a message is sent from party $\mathcal{P}_\mathbf{i}$ to the other parties, we actually execute the procedure bellow. Let $\mathbf{r}$ be a generic round of the protocol, then each party proceeds as follows when sending her messages:

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

Fig. 5: Protocol Π_{FPA} (Off-chain messages exchange).

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 3. All parties $\mathcal{P}_\mathbf{i}$ proceed as follows:

1. $\mathcal{P}_\mathbf{i}$ sends a secret deposit containing their bid $b_\mathbf{i}$, change *change* and a fee *work* to the smart contract through a confidential transaction (as described in Section 2.2). Let $Addr_\mathbf{i}$ be the address associated to party $\mathbf{i}$ and $Addr_s$ be the address associated to the smart contract, $\mathcal{P}_\mathbf{i}$ proceeds as follows:
   (a) $\mathcal{P}_\mathbf{i}$ sends (PARAM, sid) to $\mathcal{F}_{\mathsf{SC}}$, receiving (PARAM, $sid, g, h, pk_{\mathcal{C}_1}, \ldots, pk_{\mathcal{C}_m}$).
   (b) $\mathcal{P}_\mathbf{i}$ computes the bit commitments as $c_{\mathbf{ir}} = g^{b_{\mathbf{ir}}} h^{r_{\mathbf{ir}}}$ to each bit $b_{\mathbf{ir}}$ of $b_\mathbf{i}$, and the bid commitment as $c_\mathbf{i} = \prod_{\mathbf{r}=1}^{l} c_{\mathbf{ir}}^{2^{l-\mathbf{r}}} = g^{b_\mathbf{i}} h^{\sum_{\mathbf{r}=1}^{l} 2^{l-\mathbf{r}} r_{\mathbf{ir}}}$. Let $r_{b_\mathbf{i}}$ be equal to $\sum_{\mathbf{r}=1}^{l} 2^{l-\mathbf{r}} r_{\mathbf{ir}}$. Then, $c_\mathbf{i}$ can be rewritten as $c_\mathbf{i} = g^{b_\mathbf{i}} h^{r_{b_\mathbf{i}}} = \mathrm{com}(b_\mathbf{i}, r_{b_\mathbf{i}})$.
   (c) Define sets $\mathsf{In} = \{(\mathtt{id_i}, \mathtt{in_i})\}$ and $\mathsf{Out} = \{(c_\mathbf{i}, Addr_s), (work, Addr_s), (\mathrm{com}(change_\mathbf{i}, r_{change_\mathbf{i}}), Addr_\mathbf{i})\}$, where $c_\mathbf{i} = \mathrm{com}(b_\mathbf{i}, r_{b_\mathbf{i}})$ is the commitment to the bid $b_\mathbf{i}$ 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 $\mathcal{C}$, $change = \mathtt{in_i} - b_\mathbf{i} - work$ and $r_{change} \xleftarrow{\$} \mathbb{Z}_q$. Note that, in this case case, $\mathtt{in_i}$ and *work* are public, while $b_\mathbf{i}$ and *work* are private.
   (d) Compute $r_{out} = r_{b_\mathbf{i}} + r_{change_\mathbf{i}}$, 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.* $c_\mathbf{i} \cdot \mathrm{com}(change, r_{change}) \stackrel{?}{=} \mathrm{com}(\mathtt{in_i} - work, r_{out})$.
   (e) Compute proofs $(\pi_{b_\mathbf{i}}, \pi_{change})$ showing that $b_\mathbf{i}, change \in [0, 2^l - 1]$, set $\mathsf{tx_i} = (\mathtt{id}, \mathsf{In}, \mathsf{Out}, \mathsf{Sig}, r_{b_\mathbf{i}} + r_{change_\mathbf{i}}, \pi)$.
   (f) Compute the shares $(\hat{\sigma}_{\mathbf{i}1}, \ldots, \hat{\sigma}_{\mathbf{i}m}, LDEI_\mathbf{i})$ of $g^{b_\mathbf{i}}$ and $h^{r_{b_\mathbf{i}}}$ using the distribution procedure from π_{PVSS} with $pk_{\mathcal{C}_1}, \ldots, pk_{\mathcal{C}_m}$ received in step (a).
   (g) Compute $CC_\mathbf{i} \leftarrow CC((pk_{\mathcal{C}_j})_{j\in[m]}, (j)_{j\in[m]}, g, h, c_\mathbf{i}, (\hat{\sigma}_j)_{j\in[m]})$ to prove consistency among the shares $(\hat{\sigma}_{\mathbf{i}1}, \ldots, \hat{\sigma}_{\mathbf{i}m})$ and the commitment terms $g^{b_\mathbf{i}}$ and $h^{r_{b_\mathbf{i}}}$ from $c_\mathbf{i} = g^{b_\mathbf{i}} h^{r_{b_\mathbf{i}}}$.
   (h) Send (SETUP, $sid, \mathcal{P}_\mathbf{i}, \mathsf{tx_i}, pk_\mathbf{i}, (\hat{\sigma}_{\mathbf{i}1}, \ldots, \hat{\sigma}_{\mathbf{i}m}), LDEI_\mathbf{i}, CC_\mathbf{i}$) to each $\mathcal{C}_j \in \mathcal{C}$.
   (i) Upon receiving (SETUP-VERIFICATION, $sid, Sig_{\mathcal{C}_j,\mathbf{i}}$) from all $\mathcal{C}_j \in \mathcal{C}$, compute $SH1_\mathbf{i} = \mathcal{H}(\mathsf{tx_i}, pk_\mathbf{i})$ and $SH2_\mathbf{i} = \mathcal{H}((\hat{\sigma}_{\mathbf{i}1}, \ldots, \hat{\sigma}_{\mathbf{i}m}), LDEI_\mathbf{i}, CC_\mathbf{i})$ and send (SETUP, $sid, \mathcal{P}_\mathbf{i}, \mathsf{tx_i}, pk_\mathbf{i}, SH1_\mathbf{i}, SH2_\mathbf{i}, Sig_{\mathcal{C}_1,\mathbf{i}}, \ldots, Sig_{\mathcal{C}_m,\mathbf{i}}$)) 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. $\mathcal{P}_\mathbf{i}$ samples $x_{\mathbf{ir}} \xleftarrow{\$} \mathbb{Z}_q$ and computes $X_{\mathbf{ir}} = g^{x_{\mathbf{ir}}}$ for $\mathbf{r} = 1, \ldots, l$, sending $c_{\mathbf{i}1}, \cdots, c_{\mathbf{i}l}$, $X_{\mathbf{i}1}, \cdots, X_{\mathbf{i}l}$ to all other parties.
3. Upon receiving all messages $c_{j1}, \cdots, c_{jl}$, $X_{j1}, \cdots, X_{jl}$ from other parties $\mathcal{P}_j$, $\mathcal{P}_\mathbf{i}$ 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 $j \in \{1, \ldots, n\} \setminus \mathbf{i}$. If this verification fails or a message is not received, proceed to the Recovery Stage.

Fig. 6: Protocol Π_{FPA} (Stage 1).

Protocol Π_{FPA} (Stages 2 and 3)

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

1. Compute $v_{\mathbf{ir}}$ as follows:

$$v_{\mathbf{ir}} = \begin{cases} Y_{\mathbf{ir}}^{x_{\mathbf{ir}}}, & \text{if } b_{\mathbf{ir}} = 0 \\ \bar{r}_{\mathbf{ir}} \xleftarrow{\$} \mathbb{Z}_q, g^{\bar{r}_{\mathbf{ir}}}, & \text{if } b_{\mathbf{ir}} = 1 \end{cases}$$

   and generate NIZK proving that $v_{\mathbf{ir}}$ has been correctly computed $BV_{\mathbf{ir}} \leftarrow BV\{b_{\mathbf{ir}}, r_{\mathbf{ir}}, x_{\mathbf{ir}}, \bar{r}_{\mathbf{ir}} \mid (\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = c_{\mathbf{ir}} = h^{r_{\mathbf{ir}}} \wedge v_{\mathbf{ir}} = Y_{\mathbf{ir}}^{x_{\mathbf{ir}}} \wedge X_{\mathbf{ir}} = g^{x_{\mathbf{ir}}}) \vee (\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = \frac{c_{\mathbf{ir}}}{g} = h^{r_{\mathbf{ir}}} \wedge v_{\mathbf{ir}} = g^{\bar{r}_{\mathbf{ir}}})\}$, sending a message $(v_{\mathbf{ir}}, BV_{\mathbf{ir}})$ to all parties.

2. Upon receiving all messages $(v_{k\mathbf{r}}, BV_{k\mathbf{r}})$ from other parties $\mathcal{P}_k$, $\mathcal{P}_{\mathbf{i}}$ checks the proofs $BV_{k\mathbf{r}}$ for $k \in \{1,\dots,n\}\backslash\mathbf{i}$ and, if all checks pass, computes $V_{\mathbf{r}} = \prod_{k=1}^{n} v_{k\mathbf{r}}$ and then goes to Stage 3 if $V_{\mathbf{r}} \neq 1$ (at least one veto), otherwise follows the steps in Stage 2 again until the round $\mathbf{r} = l$. Note that, unless all the bids are equal to 0, at some point the condition $V_{\mathbf{r}} \neq 1$ is satisfied. If a message is not received from party $\mathcal{P}_k$ or if $BV_{k\mathbf{r}}$ is invalid, proceed to the Recovery Stage.

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

1. Compute $v_{\mathbf{ir}}$ as follows:

$$v_{\mathbf{ir}} = \begin{cases} Y_{\mathbf{ir}}^{x_{\mathbf{ir}}}, & \text{if } b_{\mathbf{ir}} = 0 \\ \bar{r}_{\mathbf{ir}} \xleftarrow{\$} \mathbb{Z}_q, g^{\bar{r}_{\mathbf{ir}}}, & \text{if } d_{\mathbf{i}\widehat{\mathbf{r}}} = 1 \wedge b_{\mathbf{ir}} = 1 \\ Y_{\mathbf{ir}}^{x_{\mathbf{ir}}}, & \text{if } d_{\mathbf{i}\widehat{\mathbf{r}}} = 0 \wedge b_{\mathbf{ir}} = 1 \end{cases}$$

   and generate NIZK proving that $v_{\mathbf{ir}}$ has been correctly computed $AV_{\mathbf{ir}} \leftarrow AV\{b_{\mathbf{ir}}, r_{\mathbf{ir}}, x_{\mathbf{ir}}, \bar{r}_{\mathbf{i}\widehat{\mathbf{r}}}, \bar{r}_{\mathbf{ir}}, x_{\mathbf{i}\widehat{\mathbf{r}}} \mid (\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = c_{\mathbf{ir}} = h^{r_{\mathbf{ir}}} \wedge v_{\mathbf{ir}} = Y_{\mathbf{ir}}^{x_{\mathbf{ir}}} \wedge X_{\mathbf{ir}} = g^{x_{\mathbf{ir}}}) \vee (\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = \frac{c_{\mathbf{ir}}}{g} = h^{r_{\mathbf{ir}}} \wedge d_{\mathbf{i}\widehat{\mathbf{r}}} = g^{\bar{r}_{\mathbf{i}\widehat{\mathbf{r}}}} \wedge v_{\mathbf{ir}} = g^{\bar{r}_{\mathbf{ir}}}) \vee (\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = \frac{c_{\mathbf{ir}}}{g} = h^{r_{\mathbf{ir}}} \wedge \ d_{\mathbf{i}\widehat{\mathbf{r}}} = Y_{\mathbf{i}\widehat{\mathbf{r}}}^{x_{\mathbf{i}\widehat{\mathbf{r}}}} \wedge X_{\mathbf{i}\widehat{\mathbf{r}}} = g^{x_{\mathbf{i}\widehat{\mathbf{r}}}} \wedge v_{\mathbf{ir}} = Y_{\mathbf{ir}}^{x_{\mathbf{ir}}} \wedge X_{\mathbf{ir}} = g^{x_{\mathbf{ir}}})\}$, sending a message $(v_{\mathbf{ir}}, AV_{\mathbf{ir}})$ to all parties.

2. Upon receiving all messages $(v_{k\mathbf{r}}, AV_{k\mathbf{r}})$ from other parties $\mathcal{P}_k$, $\mathcal{P}_{\mathbf{i}}$ checks the proofs $AV_{k\mathbf{r}}$ for $k \in \{1,\dots,n\}\backslash\mathbf{i}$ and, if all checks pass, computes $V_{\mathbf{r}} = \prod_{k=1}^{n} v_{k\mathbf{r}}$, following the steps in Stage 3 again until round $\mathbf{r} = l$. If a message is not received from party $\mathcal{P}_k$ or if $AV_{k\mathbf{r}}$ is invalid, proceed to the Recovery Stage.

Fig. 7: Protocol Π_{FPA} (Stages 2 and 3).

Protocol Π_{FPA} (Stages 4 and Recovery)

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

1. $\mathcal{P}_{\mathbf{i}}$ computes the winning bid as $b_w = b_{w1}|\cdots|b_{wl}$, such that $b_{w\mathbf{r}} = 1$ if $V_{\mathbf{r}} \neq 1$ and $b_{w\mathbf{r}} = 0$ if $V_{\mathbf{r}} = 1$, 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, $\mathcal{P}_{\mathbf{i}}$ 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 3 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 $msg_{\mathbf{ri}}$ or a signature $sig_{sk_{\mathbf{i}}}(msg_{\mathbf{r}-1,\mathbf{i}'})$, on a message $msg_{\mathbf{r}-1,\mathbf{i}'}$ by $\mathcal{P}'_{\mathbf{i}}$, expected to be sent in round $\mathbf{r}$ by $\mathcal{P}_{\mathbf{i}}$ is not received by $\mathcal{P}_k$. Then, $\mathcal{P}_k$ sends to $\mathcal{F}_{\mathsf{SC}}$ the message (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, $\mathcal{P}_{\mathbf{i}}$ is expected to send $msg_{\mathbf{ri}}$ or $sig_{sk_{\mathbf{i}}}(msg_{\mathbf{r}-1,\mathbf{i}'})$ to $\mathcal{F}_{\mathsf{SC}}$. If no action is taken, $\mathcal{P}_{\mathbf{i}}$ is identified as a cheater.

– **Conflicting messages or Invalid message**: In round $\mathbf{r}$, $\mathcal{P}_{\mathbf{i}}$ sends conflicting messages $msg_{\mathbf{ri}}$, $sig_{sk_{\mathbf{i}}}(msg_{\mathbf{ri}})$ and $msg'_{\mathbf{ri}}$, $sig_{sk_{\mathbf{i}}}(msg'_{\mathbf{ri}})$ 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 $\pi_c = (msg_{\mathbf{ri}}, sig_{sk_{\mathbf{i}}}(msg_{\mathbf{ri}}), msg'_{\mathbf{ri}}, sig_{sk_{\mathbf{i}}}(msg'_{\mathbf{ri}}))$. Otherwise, $\mathcal{P}_{\mathbf{i}}$ sends an invalid message $msg_{\mathbf{ri}}, sig_{sk_{\mathbf{i}}}(msg_{\mathbf{ri}})$ to $\mathcal{P}_k$ (*i.e.* the message does not follow the structure described in the protocol for messages in round $\mathbf{r}$), $\mathcal{P}_k$ uses this message as a proof of cheating $\pi_c = (msg_{\mathbf{ri}}, sig_{sk_{\mathbf{i}}}(msg_{\mathbf{ri}}))$. $\mathcal{P}_k$ sends (RECOVERY-CHEAT, $sid, \mathcal{P}_{\mathbf{i}}, \pi_c$) to the smart contract and $\mathcal{P}_{\mathbf{i}}$ is identified as a cheater.

Every party $\mathcal{P}_{\mathbf{i}}$ identified as a cheater loses her whole deposit ($b_{\mathbf{i}} + work$), 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 $\mathcal{P}_{\mathbf{i}} \in \mathcal{P}$ compute a NIZK $NW_{\mathbf{i}} \leftarrow NW\{x_{\mathbf{i}1}, \ldots, x_{\mathbf{i}l} \mid (V_1 = 1 \wedge v_{\mathbf{i}1} = Y_{\mathbf{i}1}^{x_{\mathbf{i}1}}) \vee \ldots \vee (V_l = 1 \wedge v_{\mathbf{i}1} = Y_{\mathbf{i}1}^{x_{\mathbf{i}1}})\}$ showing that they are not the winner. Then they send to $\mathcal{F}_{\mathsf{SC}}$ (RECOVERY-PAYMENT, $sid, NW_{\mathbf{i}}$). The winner $\mathcal{P}_w$ (in case it is identified) or all parties $\mathcal{P}_{\mathbf{i}}$ 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. 8: Protocol Π_{FPA} (Stages 4 and Recovery).

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

$\mathcal{F}_{SPA}$ 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, as well as an adversary $\mathcal{S}_{SPA}$. $\mathcal{F}_{SPA}$ is parameterized by a bid bit-length l and keeps an initially empty list bids.

- **Setup (Bid Registration):** Upon receiving $(\text{BID}, sid, \mathsf{coins}(b_\mathbf{i} + work))$ from $\mathcal{P}_\mathbf{i}$ where $b_\mathbf{i} \in \{0,1\}^l$ and $work$ is the amount required to compensate the cost of running the protocol for all the other parties, $\mathcal{F}_{SPA}$ appends $b_\mathbf{i}$ to bids.
- **Winner Selection:** After receiving $(\text{BID}, sid, \mathsf{coins}(b_\mathbf{i} + work))$ from all parties in $\mathcal{P}$, for $\mathbf{r} = 1, \ldots, \mathbf{r}_w$, where $\mathbf{r}_w$ is initialized to l, $\mathcal{F}_{SPA}$ proceeds as follows:
  1. Select $b_{w\mathbf{r}}$, *i.e.*, the $\mathbf{r}$-th bit of the highest bid b_w in the list bids.
  2. Send $(\text{ROUND-WINNER}, sid, b_{w\mathbf{r}})$ to all parties.
  3. Check if $b_{w\mathbf{r}} = 1$ and $b_{\mathbf{ir}} = 0$ for $\mathbf{i} = 1, \ldots, n \neq w$. If so, set $\mathbf{r}_w = \mathbf{r}$, that is the first position where b_w has a bit 1 and b_{w_2} has a bit 0, and send $(\text{LEAK-TO-WINNER}, sid, \mathbf{r}_w)$ to $\mathcal{P}_w$.
  4. If $\mathcal{P}_w$ is honest, announce her identity by sending $(\text{ANNOUNCE-WINNER}, sid, \mathcal{P}_w)$ to all $\mathcal{P}_i \in \mathcal{P}$ and $\mathcal{S}_{SPA}$. If $\mathcal{P}_w$ is corrupted, send $(\text{ANNOUNCE?}, sid, \mathcal{P}_w)$ to $\mathcal{S}_{SPA}$. If $\mathcal{S}_{SPA}$ answers with $(\text{ANNOUNCE}, sid)$ then send $(\text{ANNOUNCE-WINNER}, sid, \mathcal{P}_w)$ to all $\mathcal{P}_i \in \mathcal{P}$. If $\mathcal{S}_{SPA}$ answers with $(\text{NOT-ANNOUNCE}, sid)$, do nothing.
  5. Send $(\text{ABORT?}, sid)$ to $\mathcal{S}_{SPA}$. If $\mathcal{S}_{SPA}$ answers with $(\text{ABORT}, sid, \mathcal{P}_\mathbf{i})$ where $\mathcal{P}_\mathbf{i}$ is corrupted, remove $b_\mathbf{i}$ from bids, remove $\mathcal{P}_\mathbf{i}$ from $\mathcal{P}$, send $(\text{ABORT}, sid, \mathcal{P}_\mathbf{i}, \mathsf{coins}(\frac{b_\mathbf{i} + work}{|\mathcal{P}|} + work))$ where $|\mathcal{P}|$ is the number of remaining parties to all other parties in $\mathcal{P}$ to $\mathcal{S}_{SPA}$, set again $\mathbf{r} = 1$ and go to Step 1. If $\mathcal{S}_{SPA}$ answers with $(\text{PROCEED}, sid)$, if $\mathcal{P}_w$ has been determined, *i.e.*, $b_{w\mathbf{r}} = 1$ and $b_{\mathbf{ir}} = 0$ for $\mathbf{i} = 1, \ldots, n \neq w$, go to Second Price Selection, otherwise increment $\mathbf{r}$ by 1 and go to Step 1.
- **Second Price Selection:** remove $\mathcal{P}_w$ from $\mathcal{P}$ and for $\mathbf{r} = \mathbf{r}_w, \ldots, l$ $\mathcal{F}_{SPA}$ proceeds as follows:
  1. Select $b_{w_2\mathbf{r}}$, *i.e.*, the $\mathbf{r}$-th bit of the second highest bid b_{w_2} in the list bids.
  2. Send $(\text{ROUND-WINNER}, sid, b_{w_2\mathbf{r}})$ to all parties.
  3. Check if $b_{w_2\mathbf{r}} = 1$ and $b_{\mathbf{ir}} = 0$ for $\mathbf{i} = 1, \ldots, n \neq w_2$. If so, set $\mathbf{r}_{w_2} = \mathbf{r}$, that is the first position where b_{w_2} has a bit 1 and b_{w_3} has a bit 0, and send $(\text{LEAK-TO-SECOND}, sid, \mathbf{r}_{w_2})$ to $\mathcal{P}_{w_2}$.
  4. Send $(\text{ABORT?}, sid)$ to $\mathcal{S}_{SPA}$. If $\mathcal{S}_{SPA}$ answers with $(\text{ABORT}, sid, \mathcal{P}_\mathbf{i})$ where $\mathcal{P}_\mathbf{i}$ is corrupted, remove $b_\mathbf{i}$ from bids, remove $\mathcal{P}_\mathbf{i}$ from $\mathcal{P}$, send $(\text{ABORT}, sid, \mathcal{P}_\mathbf{i}, \mathsf{coins}(\frac{b_\mathbf{i} + work}{|\mathcal{P}|}))$ where $|\mathcal{P}|$ is the number of remaining parties to all other parties in $\mathcal{P}$, set again $\mathbf{r} = \mathbf{r}_w$ and go to Step 1 to start recomputing the second price. If $\mathcal{S}_{SPA}$ answers with $(\text{PROCEED}, sid)$, if $\mathcal{P}_{w_2}$ has been determined, *i.e.*, $b_{w_2\mathbf{r}} = 1$ and $b_{\mathbf{ir}} = 0$ for $\mathbf{i} = 1, \ldots, n \neq w_2$, go to Payout, otherwise increment $\mathbf{r}$ by 1 and go to Step 1.
- **Payout:** Send $(\text{REFUND}, sid, \mathsf{coins}(b_\mathbf{i} + work))$ to all parties $\mathcal{P}_\mathbf{i} \neq \mathcal{P}_w$, send $(\text{REFUND}, sid, \mathsf{coins}(work))$ to $\mathcal{P}_w$, send $\mathsf{coins}(b_{w_2})$ to $\mathcal{P}_{AUC}$, and halt.

Fig. 9: Functionality $\mathcal{F}_{SPA}$.

5 Extension to Second price Auctions

The second price sealed bid auction is a type of auction in which the parties first submit their bids to the auctioneer and then the winner is the party with the highest bid, however the price she pays is the second highest bid. The importance of the second price auction is that it is *strategy proof*. That is, the best strategy for rational parties is to bid their true valuation of the auctioned good. Despite this, the sealed second price auctions may not be applied applied in certain scenarios due to the trust that has to be put in the auctioneer. In particular, a dishonest auctioneer may manipulate the bids and substitute the second highest bid with a bid that is slightly smaller than the first bid, so as to increase her revenue, or disclose the losing bids of the other parties to have a financial gain. In fact, in a recent study [2] it is shown that the only auction in which the auctioneer has no incentive to deviate from the rules is the first price auction. Hence, when considering the second price we must overcome these problems.

Modelling Second Price Fair Auctions: First we describe an ideal functionality $\mathcal{F}_{SPA}$ for the second price auctions we realize in Figure 9.

The Protocol: Protocol Π_{SPA} for Second Price Auctions is described in Figures 10 and 11. In this protocol, parties to check if they are the only one veto-ing in every round $\mathbf{r}$ where $V_{\mathbf{r}} \neq 1$ (*i.e.* where there was a veto), which means that they are the winning party with the highest bid. Parties can do that by checking whether they obtain alternative value $V'_{\mathbf{r}} = 1$ if they compute it as they would compute $V_{\mathbf{r}}$ but using with an alternative value $v'_{\mathbf{ir}} = Y_{\mathbf{ir}}^{x_{\mathbf{ir}}}$ (no veto) for their messages and keeping the other parties' messages. If this condition is satisfied, then party $\mathcal{P}_{\mathbf{i}}$ proves it to all the other parties by revealing $x_{\mathbf{ir}}$. The first party who proves this condition to be true is the winning party $\mathcal{P}_w$. In order to compute the second highest bid b_{w_2}, the other parties conclude the protocol excluding $\mathcal{P}_w$'s inputs. Note that, by using this approach the winning bid is just partially disclosed, *i.e.*, the knowledge of the round $\mathbf{r}$ in which $\mathcal{P}_w$ declared herself as the winner of the auctions provides a lower bound only over her actual bid b_w. Moreover, the parties do not need to run the entire protocol again, which reduces drastically both computational and communication complexity of the protocol with respect to the naive solution of re-executing the protocol from scratch without the winner's bid. In particular, the complexity of the protocol for Second Price Sealed Bid Auctions is almost the same as the one for the First Price case. Indeed, when $\mathcal{P}_w$ sends $x_{\mathbf{ir}}$ to all the other parties, so as to prove she is the winning party, the communication complexity increases by $|\mathbb{Z}_q|$. Moreover, when a winner is identified, only one round has to be re-executed without the winner's bid. We present a detailed efficiency estimate in Section 6.

Security Analysis: The security of Protocol Π_{SPA} is stated in Theorem 2 and proven in Appendix E. A game theoretical analysis is presented in Appendix F.

Theorem 2. *Under the DDH Assumption, Protocol Π_{SPA} securely computes $\mathcal{F}_{SPA}$ 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 $\mathcal{C}_i \in \mathcal{C}$.*

Protocol Π_{SPA} (Stages 1, 2, 3a and 3b)

Protocol Π_{SPA} is executed by parties $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ with a l-bit bids $b_{\mathbf{i}} = b_{\mathbf{i}1}|\ldots|b_{\mathbf{i}l}$ and a deposit committee $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$, interacting among themselves and with a smart contract $\mathcal{F}_{\mathsf{SC}}$.

Off-chain messages exchange and Stage 1 - Setup: Same as in Protocol Π_{FPA}.

Stage 2 - Before First Veto and Stage 3a - After First Veto: In this stage, all parties $\mathcal{P}_{\mathbf{i}}$ follow all the steps of Stage 2 and Stage 3 of Π_{FPA} respectively and execute the following extra steps:

1. If $\mathcal{P}_{\mathbf{i}}$ sent $v_{\mathbf{ir}} = g^{\bar{r}_{\mathbf{ir}}}$ (*i.e.*, she has $b_{\mathbf{ir}} = 1$ and issued a veto), $\mathcal{P}_{\mathbf{i}}$ computes $V'_{\mathbf{r}} = \big(\prod_{k=1,k\neq i}^{n} v_{k\mathbf{r}}\big) Y_{\mathbf{ir}}^{x_{\mathbf{ir}}}$ and checks whether $V_{\mathbf{r}} \neq 1$ and $V'_{\mathbf{r}} = 1$. If this is true, it means that $\mathcal{P}_{\mathbf{i}}$ is the only one who has vetoed (*i.e.*, she is the only party with $b_{k\mathbf{r}} = 1$ for $k = 1, \ldots, n$, implying she has the highest bid). In this case, $\mathcal{P}_{\mathbf{i}}$ sends (WINNER, $sid, \mathcal{P}_w, x_{\mathbf{ir}}$) to all other parties.
2. Upon receiving (WINNER, $sid, \mathcal{P}_w, x_{\mathbf{ir}}$) from $\mathcal{P}_w$, $\mathcal{P}_i$ it checks whether $\mathcal{P}_w$ indeed has the highest bid by checking that $X_{w\mathbf{r}} = g^{x_{w\mathbf{r}}}$, computing $V'_{\mathbf{r}} = \big(\prod_{k=1,k\neq w}^{n} v_{w\mathbf{r}}\big) Y_{w\mathbf{r}}^{x_{w\mathbf{r}}}$ and checking whether $V_{\mathbf{r}} \neq 1$ and $V'_{\mathbf{r}} = 1$. If any of these checks fail, proceed to the Recovery Stage.
3. If a valid message (WINNER, $sid, \mathcal{P}_w, x_{\mathbf{ir}}$) was $\mathcal{P}_w$, all parties $\mathcal{P}_{\mathbf{i}} \in \mathcal{P} \setminus \mathcal{P}_w$ recompute $Y'_{k\mathbf{r}} = \prod_{m=1}^{k-1} X_{m\mathbf{r}} / \prod_{m=k+1}^{n} X_{m\mathbf{r}}$ for $k \in \{1, \ldots, n\} \setminus w$, $k = j+1, \ldots, l$ (*i.e.*, excluding $\mathcal{P}_w$ from the remaining rounds from $j+1$ to l), then continue to Stage 3b by executing the protocol using the new $Y'_{\mathbf{ir}}$ with a set of parties $\mathcal{P} \setminus \mathcal{P}_w$ excluding $\mathcal{P}_w$.
4. If no party has sent a message (WINNER, $sid, \mathcal{P}_w, x_{\mathbf{ir}}$) by round $\mathbf{r} = l$, then the winner is dishonest (assuming a tie is not possible) and $\mathcal{P}_i$ proceeds to the Recovery Stage.

Stage 3b - After First Unique Veto: In this stage, all parties $\mathcal{P}_{\mathbf{i}} \in \{\mathcal{P}_1, \ldots, \mathcal{P}_n\} \setminus \mathcal{P}_w$ considers $V_{\mathbf{r}} = V'_{\mathbf{r}} = \big(\prod_{k=1,k\neq w}^{n} v_{w\mathbf{r}}\big) Y_{w\mathbf{r}}^{x_{w\mathbf{r}}} = 1$ (*i.e.*, $V_{\mathbf{r}}$ with the input $v_{w\mathbf{r}}$ of $\mathcal{P}_w$ not representing a veto instead). Then:

- If there does not exist another previous round z with $z = 1, \ldots, \mathbf{r}-1$ such that $V_z \neq 1$, *i.e.*, the first veto was also the first unique veto, then the parties $\mathcal{P}_{\mathbf{i}} \in \{\mathcal{P}_1, \ldots, \mathcal{P}_n\} \setminus \mathcal{P}_w$ continue the protocol following all the steps is Stage 2 and Stage 3 described in Section 4 but using the values $Y'_{\mathbf{ir}}$ recomputed after the first unique veto was detected. When Stage 2 and eventually Stage 3 are completed, return and go to the output stage described in this section.
- If there does exist another previous round z with $z = 1, \ldots, \mathbf{r}-1$ such that $V_z \neq 1$, *i.e.*, the first unique veto was not the first veto, then set $\hat{\mathbf{r}} = z$ (*i.e.*, the index of the last veto is changed from $\mathbf{r}$ to z) and the parties $\mathcal{P}_{\mathbf{i}} \in \{\mathcal{P}_1, \ldots, \mathcal{P}_n\} \setminus \mathcal{P}_w$ continue the protocol following all the steps from Stage 3 as described in Section 4 but using $\hat{\mathbf{r}} = z$ when computing their next input $v_{\mathbf{i}(\mathbf{r}+1)}$ and the values $Y'_{\mathbf{ir}}$ recomputed after the first unique veto was detected. When Stage 3 is completed, return and go to the output stage described in this section.

Fig. 10: Protocol Π_{SPA} (Stages 1, 2, 3a and 3b).

Protocol Π_{SPA} (Stage 4 and Recovery Stage)

Stage 4 - Output: At this point, all parties know the winner party $\mathcal{P}_w$ and the second price b_{w_2}. The protocol proceeds as follows:

1. $\mathcal{P}_\mathbf{i}$ computes the second highest bid as $b_{w_2} = b_{w_2 1}|\cdots|b_{w_2 l}$, such that $b_{w_2 \mathbf{r}} = 1$ if $V_\mathbf{r} \neq 1$ and $b_{w_2 \mathbf{r}} = 0$ if $V_\mathbf{r} = 1$, and sends $\mathcal{P}_w, b_{w_2}$ to all other parties (causing all parties $\mathcal{P}_k$ to sign b_{w_2} and send $sig_{sk_k}(\mathcal{P}_w|b_{w_2})$ to each other).
2. In the Setup Stage, $\mathcal{P}_w$ sent to the smart contract a confidential transaction $\mathsf{tx}_w = (\mathtt{id}, \mathsf{In}, \mathsf{Out}, \mathsf{Sig}, r_{b_w} + r_{change_w}, \pi)$ where $\mathsf{Out} = \{(\mathrm{com}(b_w, r_{b_w}), Addr_s), (work, Addr_s), (\mathrm{com}(change_w, r_{change_w}), Addr_w)\}$. $\mathcal{P}_w$ creates a new confidential transaction $\mathsf{tx}_{pay} = (\mathtt{id}_{pay}, \mathsf{In}_{pay}, \mathsf{Out}_{pay}, \mathsf{Sig}_{pay}, r_{b_w} - r_{change'_w}, \pi_{pay})$ where $\mathsf{In} = \{(\mathtt{id}, \mathrm{com}(b_w, r_{b_w}))\}$, $\mathsf{Out} = \{(b_{w_2}, Addr_{auc}), (\mathrm{com}(change'_w, r_{change'_w}), Addr_w)\}$, Sig_{pay} is left empty, $Addr_{auc}$ is the address of the auctioneer, $Addr_w$ is the address of $\mathcal{P}_w$ and π_{pay} is a NIZK showing that $change'_w$ is between $[0, 2^l - 1]$. $\mathcal{P}_w$ sends (OUTPUT, $sid, \mathcal{P}_w, \mathsf{tx}_{pay}, \{sig_{sk_k}(\mathcal{P}_w|b_{w_2})\}_{k\in[n]}$) to $\mathcal{F}_{\mathsf{SC}}$, which performs tx_{pay} after checking the validity of the message so that the auctioneer receives the payment b_{w_2} and $\mathcal{P}_w$ gets back $b_w - b_{w_2}$.
3. Finally, all honest parties receive a refund of their deposit from the smart contract, apart from the winning party, who only receives a refund of the fee $work$ plus the transaction $(b_w - b_{w_2})$ computed in the previous step.

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 3 if requested. In case a party $\mathcal{P}_i \in \mathcal{P}$ is suspected of cheating, the Recovery stage is executed depending on the exact suspected cheating as defined in Protocol Π_{FPA}, which allows to eventually identify the cheater. If a cheater $\mathcal{P}_\mathbf{i}$ is identified, the Recovery Stage proceeds as follows:

- **Re-execution (unknown $\mathcal{P}_w$)**: in this scenario the winning the party $\mathcal{P}_w$ is still unknown, then the parties $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\} \setminus \mathcal{P}_\mathbf{i}$ re-execute the protocol from Stage 2 without the cheating party $\mathcal{P}_\mathbf{i}$.
- **Re-execution (known $\mathcal{P}_w$ but unknown b_{w_2})**: in this scenario the winning party $\mathcal{P}_w$ is known but the second highest bid b_{w_2} is unknown, then the parties $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\} \setminus \{\mathcal{P}_\mathbf{i}, \mathcal{P}_w\}$ re-execute the protocol from Stage 2 without the cheating party $\mathcal{P}_\mathbf{i}$ and the winning party $\mathcal{P}_w$.
- **Complete payment (known $\mathcal{P}_w$ and b_{w_2} but missing payment)**: in this scenario both the winning party $\mathcal{P}_w$ and the second highest bid b_{w_2} are known, but $\mathcal{P}_w$ is has not completed the payment to the auctioneer. Then, $\mathcal{P}_w$'s deposit $b_w + work$ is distributed by the smart contract as follows: b_{w_2} is sent to the auctioneer and the remaining amount, that is equal to $(b_w + work - b_{w_2})$, is distributed to the other parties.
- **Dishonest winner identification:** If no party has sent a message (WINNER, $sid, \mathcal{P}_w, x_{\mathbf{i}\mathbf{r}}$) by round $\mathbf{r} = l$, each $\mathcal{P}_\mathbf{i} \in \mathcal{P}$ computes a NIZK $NW_\mathbf{i} \leftarrow NW\{x_{\mathbf{i}1}, \ldots, x_{\mathbf{i}l} \mid (V_1 = 1 \wedge v_{\mathbf{i}1} = Y_{\mathbf{i}1}^{x_{\mathbf{i}1}}) \vee \ldots \vee (V_l = 1 \wedge v_{\mathbf{i}1} = Y_{\mathbf{i}1}^{x_{\mathbf{i}1}})\}$ showing that they are not the winner and sends to $\mathcal{F}_{\mathsf{SC}}$ (RECOVERY-DISHONEST-WINNER, $sid, NW_\mathbf{i}$). All parties $\mathcal{P}_\mathbf{i}$ who do not send a valid NW_i are identified as dishonest and have their deposits distributed among the honest parties.

Fig. 11: Protocol Π_{SPA} (Stage 4 and Recovery Stage).

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 [4]. In the second price case, we show that Π_{SPA} only incurs a small overhead (dominated by re-executing one round) over Π_{FPA}.

	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 [4]	$11l + 12nl$	$\tau(17 + 20n)$	$(l - \tau)(33 + 36n)$	$48nl + 44l - 16\tau - 16n\tau$

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

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, $\lvert\mathbb{G}\rvert$ and $\lvert\mathbb{Z}_q\rvert$ 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 Section 2.

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 [4], 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 has been expressed in terms of $\lvert\mathbb{G}\rvert$. 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 the communication of the proposed second price auction is still linear in the number of agents. That is, assuming that at round $\mathbf{r}$, there is a party who is the only one that is veto-ing, then the parties have to re-run the $\mathbf{r}^{th}$ 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. Masayuki Abe and Koutarou Suzuki. M+1-st price auction using homomorphic encryption. In David Naccache and Pascal Paillier, editors, *PKC 2002*, volume 2274 of *LNCS*, pages 115–124. Springer, Heidelberg, February 2002.
2. Mohammad Akbarpour and Shengwu Li. Credible auctions: A trilemma. *Econometrica*, 88(2):425–467, 2020.
3. Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. Secure multiparty computations on bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 443–458. IEEE Computer Society Press, May 2014.
4. Samiran Bag, Feng Hao, Siamak F Shahandashti, and Indranil G Ray. Seal: Sealed-bid auction without auctioneers. *IEEE Transactions on Information Forensics and Security*, 2019.
5. Olivier Baudron and Jacques Stern. Non-interactive private auctions. In Paul F. Syverson, editor, *FC 2001*, volume 2339 of *LNCS*, pages 364–378. Springer, Heidelberg, February 2002.
6. Carsten Baum, Bernardo David, and Rafael Dowsley. A framework for universally composable publicly verifiable cryptographic protocols. *IACR Cryptol. ePrint Arch.*, 2020:207, 2020.
7. Carsten Baum, Bernardo David, and Rafael Dowsley. Insured MPC: Efficient secure computation with financial penalties. In Joseph Bonneau and Nadia Heninger, editors, *FC 2020*, volume 12059 of *LNCS*, pages 404–420. Springer, Heidelberg, February 2020.
8. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi

Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.
9. F. Benhamouda, S. Halevi, and T. Halevi. Supporting private data on hyperledger fabric with secure multiparty computation. In *2018 IEEE International Conference on Cloud Engineering (IC2E)*, pages 357–363, April 2018.
10. Fabrice Benhamouda, Craig Gentry, Sergey Gorbunov, Shai Halevi, Hugo Krawczyk, Chengyu Lin, Tal Rabin, and Leonid Reyzin. Can a public blockchain keep a secret? In *TCC 2020, Part I*, LNCS, pages 260–290. Springer, Heidelberg, March 2020.
11. Iddo Bentov and Ranjit Kumaresan. How to use bitcoin to design fair protocols. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 421–439. Springer, Heidelberg, August 2014.
12. Iddo Bentov, Ranjit Kumaresan, and Andrew Miller. Instantaneous decentralized poker. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 410–440. Springer, Heidelberg, December 2017.
13. Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *Journal of Cryptographic Engineering*, 2(2):77–89, September 2012.
14. Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In Roger Dingledine and Philippe Golle, editors, *FC 2009*, volume 5628 of *LNCS*, pages 325–343. Springer, Heidelberg, February 2009.
15. Peter Bogetoft, Ivan Damgård, Thomas Jakobsen, Kurt Nielsen, Jakob Pagter, and Tomas Toft. A practical implementation of secure auctions based on multiparty integer computation. In Giovanni Di Crescenzo and Avi Rubin, editors, *FC 2006*, volume 4107 of *LNCS*, pages 142–147. Springer, Heidelberg, February / March 2006.
16. Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. SoK: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy*, pages 104–121. IEEE Computer Society Press, May 2015.
17. Felix Brandt. Secure and private auctions without auctioneers. *Technical Report FKI-245–02. Institut fur Informatick, Technishce Universitat Munchen*, 2002.
18. Felix Brandt. Fully private auctions in a constant number of rounds. In Rebecca Wright, editor, *FC 2003*, volume 2742 of *LNCS*, pages 223–238. Springer, Heidelberg, January 2003.
19. Felix Brandt. How to obtain full privacy in auctions. *International Journal of Information Security*, 5(4):201–216, 2006.
20. Felix Brandt and Tuomas Sandholm. Efficient privacy-preserving protocols for multi-unit auctions. In Andrew Patrick and Moti Yung, editors, *FC 2005*, volume 3570 of *LNCS*, pages 298–312. Springer, Heidelberg, February / March 2005.
21. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.
22. Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. *Technical Report/ETH Zurich, Department of Computer Science*, 260, 1997.

23. Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, January 2000.
24. Ignacio Cascudo and Bernardo David. ALBATROSS: Publicly AttestabLe BATched Randomness based On Secret Sharing. In *ASIACRYPT 2020, Part III*, LNCS, pages 311–341. Springer, Heidelberg, December 2020.
25. David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, Heidelberg, August 1993.
26. Jing Chen and Silvio Micali. Algorand: A secure and efficient distributed ledger. *Theor. Comput. Sci.*, 777:155–183, 2019.
27. Arka Rai Choudhuri, Matthew Green, Abhishek Jain, Gabriel Kaptchuk, and Ian Miers. Fairness in an unfair world: Fair multiparty computation from public bulletin boards. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 719–728. ACM Press, October / November 2017.
28. Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *18th ACM STOC*, pages 364–369. ACM Press, May 1986.
29. Peter Cramton et al. Spectrum auctions. *Handbook of telecommunications economics*, 1:605–639, 2002.
30. Bernardo David, Rafael Dowsley, and Mario Larangeira. Kaleidoscope: An efficient poker protocol with payment distribution and penalty enforcement. In Sarah Meiklejohn and Kazue Sako, editors, *FC 2018*, volume 10957 of *LNCS*, pages 500–519. Springer, Heidelberg, February / March 2018.
31. Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 66–98. Springer, Heidelberg, April / May 2018.
32. Jannik Dreier, Jean-Guillaume Dumas, and Pascal Lafourcade. Brandt's fully private auction protocol revisited. *Journal of Computer Security*, 23(5):587–610, 2015.
33. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.
34. Matthew K Franklin and Michael K Reiter. The design and implementation of a secure auction service. *IEEE Transactions on Software Engineering*, 22(5):302–312, 1996.
35. Hisham S. Galal and Amr M. Youssef. Trustee: Full privacy preserving vickrey auction on top of Ethereum. In Andrea Bracciali, Jeremy Clark, Federico Pintore, Peter B. Rønne, and Massimiliano Sala, editors, *FC 2019 Workshops*, volume 11599 of *LNCS*, pages 190–207. Springer, Heidelberg, February 2019.
36. Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.
37. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 171–185. Springer, Heidelberg, August 1987.
38. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

39. Feng Hao and Piotr Zieliński. A 2-round anonymous veto protocol. In *International Workshop on Security Protocols*, pages 202–211. Springer, 2006.
40. Michael Harkavy, J Doug Tygar, and Hiroaki Kikuchi. Electronic auctions with private bids. In *USENIX Workshop on Electronic Commerce*, 1998.
41. Yuval Ishai, Rafail Ostrovsky, and Vassilis Zikas. Secure multi-party computation with identifiable abort. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 369–386. Springer, Heidelberg, August 2014.
42. Ari Juels and Michael Szydlo. A two-server, sealed-bid auction protocol. In Matt Blaze, editor, *FC 2002*, volume 2357 of *LNCS*, pages 72–86. Springer, Heidelberg, March 2003.
43. Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 357–388. Springer, Heidelberg, August 2017.
44. Aggelos Kiayias, Hong-Sheng Zhou, and Vassilis Zikas. Fair and robust multi-party computation using a global transaction ledger. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 705–734. Springer, Heidelberg, May 2016.
45. Paul Klemperer. *Auctions: theory and practice*. Princeton University Press, 2004.
46. Vijay Krishna. *Auction theory*. Academic press, 2009.
47. Ranjit Kumaresan and Iddo Bentov. Amortizing secure computation with penalties. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 418–429. ACM Press, October 2016.
48. Ranjit Kumaresan, Tal Moran, and Iddo Bentov. How to use bitcoin to play decentralized poker. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 195–206. ACM Press, October 2015.
49. Ranjit Kumaresan, Vinod Vaikuntanathan, and Prashant Nalini Vasudevan. Improvements to secure computation with penalties. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 406–417. ACM Press, October 2016.
50. Kaoru Kurosawa and Wakaha Ogata. Bit-slice auction circuit. In *European Symposium on Research in Computer Security*, pages 24–38. Springer, 2002.
51. Helger Lipmaa, N. Asokan, and Valtteri Niemi. Secure Vickrey auctions without threshold trust. In Matt Blaze, editor, *FC 2002*, volume 2357 of *LNCS*, pages 87–101. Springer, Heidelberg, March 2003.
52. Andreu Mas-Colell, Michael Dennis Whinston, Jerry R Green, et al. *Microeconomic theory*, volume 1. Oxford university press New York, 1995.
53. Greg Maxwell. Confidential transactions. https://people.xiph.org/ greg/confidential_values.txt, 2016.
54. Peter Bro Miltersen, Jesper Buus Nielsen, and Nikos Triandopoulos. Privacy-enhancing auctions using rational cryptography. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 541–558. Springer, Heidelberg, August 2009.
55. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.
56. Hannu Nurmi and Arto Salomaa. Cryptographic protocols for vickrey auctions. *Group Decision and Negotiation*, 2(4):363–373, 1993.
57. Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. In Andréa W. Richa, editor, *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, volume 91

of *LIPIcs*, pages 39:1–39:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.
58. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.
59. David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398. Springer, Heidelberg, May 1996.
60. William Thomson. *How to Divide When There Isn't Enough.* Number 62. Cambridge University Press, 2019.
61. Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the keys to the intel SGX kingdom with transient out-of-order execution. In William Enck and Adrienne Porter Felt, editors, *USENIX Security 2018*, pages 991–1008. USENIX Association, August 2018.
62. William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance*, 16(1):8–37, 1961.

Appendix

A Non-interactive Zero Knowledge Proofs of Knowledge for Stages 2 and 3

We employ the approach of Camenisch and Stadler [22] based on the Fiat-Shamir heuristc [33] in order to obtain non interactive zero knowledge (NIZK) proofs of knowledge for discrete logarithm relations. In such NIZK proof systems, no interaction is required between the prover and the verifier. This property is crucial for obtaining public verifiability in our protocol, which we need in order to implement our punishment mechanism.

In particular, we are interested in proving knowledge of the discrete logarithm of h base g (denoted by $DL\{w \mid h = g^w\}$), proving knowledge of discrete logarithms of h_1 base g_1 and h_2 base g_2 that satisfy a linear equation $a_1w_1+a_2w_2 = b$ (denoted by $DLEQ\{w_1, w_2 \mid h_1 = g_1^{w_1} \wedge h_2 = g_2^{w_2} \wedge a_1w_1 + a_2w_2 = b\}$) and proving knowledge of either the discrete logarithm of h_1 base g_1 or h_2 base g_2, without disclosing which one (denoted by $DLOR\{w_1, w_2 \mid (h_1 = g_1^{w_1}) \vee (h_2 = g_2^{w_2})\}$). We combine such simple statements using the approach of [22] in order to prove more complicated relations that are used in forcing parties to execute our protocols correctly using the GMW methodology [37].

A.1 NIZK for Stage 2 - Before First Veto

In this section we show the structure of proofs for the required NIZKs of Stage 2. The NIZKs prove knowledge of either $b_{\mathbf{ir}}, r_{\mathbf{ir}}, x_{\mathbf{ir}}$ such that $\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = c_{\mathbf{ir}} = h^{r_{\mathbf{ir}}} \wedge v_{\mathbf{ir}} = Y_{\mathbf{ir}}^{x_{\mathbf{ir}}} \wedge X_{\mathbf{ir}} = g^{x_{\mathbf{ir}}}$ or of $b_{\mathbf{ir}}, r_{\mathbf{ir}}, \bar{r}_{\mathbf{ir}}$ such that $\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = \frac{c_{\mathbf{ir}}}{g} = h^{r_{\mathbf{ir}}} \wedge v_{\mathbf{ir}} = g^{\bar{r}_{\mathbf{ir}}}$, where $v_{\mathbf{ir}}, c_{\mathbf{ir}}, X_{\mathbf{ir}}, g, Y_{\mathbf{ir}}$ are public. We denote this NIZK by

$$BV\{b_{\mathbf{ir}}, r_{\mathbf{ir}}, x_{\mathbf{ir}}, \bar{r}_{\mathbf{ir}} \,|(\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = c_{\mathbf{ir}} = h^{r_{\mathbf{ir}}} \wedge v_{\mathbf{ir}} = Y_{\mathbf{ir}}^{x_{\mathbf{ir}}} \wedge X_{\mathbf{ir}} = g^{x_{\mathbf{ir}}}) \vee$$
$$(\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = \frac{c_{\mathbf{ir}}}{g} = h^{r_{\mathbf{ir}}} \wedge v_{\mathbf{ir}} = g^{\bar{r}_{\mathbf{ir}}})\}$$

Following the approach proposed by Camenisch and Stadler [22] we construct this NIZK as follows:

$$\tilde{F}_1 = DL(h, c_{\mathbf{ir}}) \otimes \Big[DL(Y_{\mathbf{ir}}, v_{\mathbf{ir}}) \cap DL(g, X_{\mathbf{ir}})\Big]$$

$$\tilde{F}_2 = DL(h, c_{\mathbf{ir}}/g) \otimes DL(g, v_{\mathbf{ir}})$$

Therefore we need to show

$$\tilde{F} = \tilde{F}_1 \cup \tilde{F}_2$$

To prove the knowledge of either $\tilde{F}_1$ or $\tilde{F}_2$, assuming that $\tilde{F}_\alpha$ is known, party $\mathbf{i}$ proceeds as follows:

1. Choose $\bar{V} = (\bar{v}_1, \bar{v}_2, \bar{v}_3, \bar{v}_4)$ with $\bar{v}_i \xleftarrow{\$} \mathbb{Z}_q$ and $\bar{w} = (\bar{w}_1, \bar{w}_2)$ with $\bar{w}_\alpha = 0$ and $\bar{w}_i \xleftarrow{\$} \mathbb{Z}_q$ for $i \neq \alpha$, and compute $t_1 = c_{\mathbf{ir}}^{\bar{w}_1} h^{\bar{v}_1}$, $t_2 = v_{\mathbf{ir}}^{\bar{w}_1} Y_{\mathbf{ir}}^{\bar{v}_2}$, $t_3 = X_{\mathbf{ir}}^{\bar{w}_1} g^{\bar{v}_2}$, $t_4 = (\frac{c_{\mathbf{ir}}}{g})^{\bar{w}_2} h^{\bar{v}_3}$, and $t_5 = v_{\mathbf{ir}}^{\bar{w}_2} g^{\bar{v}_4}$
2. $H = \mathcal{H}(h, c_{\mathbf{ir}}, Y_{\mathbf{ir}}, v_{\mathbf{ir}}, g, X_{\mathbf{ir}}, \frac{c_{\mathbf{ir}}}{g}, t_1, t_2, t_3, t_4, t_5) \pmod q$.
3. $\Gamma = (\gamma_1, \gamma_2)$ where

$$\gamma_i = \begin{cases} H - (\bar{w}_1 + \bar{w}_2) \pmod q, & \textit{if} \quad i = \alpha \\ \bar{w}_i, & \textit{otherwise} \end{cases}$$

4. $R = (r_1, r_2, r_3, r_4, r_5)$ where $r_1 = \bar{v}_1 - \gamma_\alpha x_1$, $r_2 = \bar{v}_2 - \gamma_\alpha x_2$, $r_3 = \bar{v}_2 - \gamma_\alpha x_2$, $r_4 = \bar{v}_3 - \gamma_\alpha x_3$, and $r_5 = \bar{v}_4 - \gamma_\alpha x_4$ (all equations are modulo q), in which $(x_1, x_2, x_3, x_4) = (r_{\mathbf{ir}}, x_{\mathbf{ir}}, 0, 0)$ if $\alpha = 1$, and $(x_1, x_2, x_3, x_4) = (0, 0, r_{\mathbf{ir}}, \bar{r}_{\mathbf{ir}})$ if $\alpha = 2$.

The resulting proof is $(\gamma_1, \gamma_2, r_1, r_2, r_3, r_4, r_5)$. The validity of the proof can be checked by first re-constructing the commitments. That is,

$$t'_1 = c_{\mathbf{ir}}^{\gamma_1} h^{r_1} \qquad t'_2 = v_{\mathbf{ir}}^{\gamma_1} Y^{r_2} \qquad t'_3 = X_{\mathbf{ir}}^{\gamma_1} g^{r_3}$$
$$t'_4 = (\frac{c_{\mathbf{ir}}}{g})^{\gamma_2} h^{r_4} \qquad t'_5 = v_{\mathbf{ir}}^{\gamma_2} g^{r_5}$$

and then checks the following condition

$$\gamma_1 + \gamma_2 = \mathcal{H}(h, c_{\mathbf{ir}}, Y_{\mathbf{ir}}, v_{\mathbf{ir}}, g, X_{\mathbf{ir}}, \frac{c_{\mathbf{ir}}}{g}, t'_1, t'_2, t'_3, t'_4, t'_5)$$

A.2 NIZK Stage 3 - After First Veto

In this section we show the structure of proofs for the required NIZKs of Stage 3. The NIZKs prove knowledge of either $b_{\mathbf{ir}}, r_{\mathbf{ir}}, x_{\mathbf{ir}}$ such that $\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = c_{\mathbf{ir}} = h^{r_{\mathbf{ir}}} \wedge v_{\mathbf{ir}} = Y_{\mathbf{ir}}^{x_{\mathbf{ir}}} \wedge X_{\mathbf{ir}} = g^{x_{\mathbf{ir}}}$, or of $b_{\mathbf{ir}}, r_{\mathbf{ir}}, \bar{r}_{\mathbf{i\hat{r}}}, \bar{r}_{\mathbf{ir}}$ such that $\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = \frac{c_{\mathbf{ir}}}{g} = h^{r_{\mathbf{ir}}} \wedge d_{\mathbf{i\hat{r}}} = g^{\bar{r}_{\mathbf{i\hat{r}}}} \wedge v_{\mathbf{ir}} = g^{\bar{r}_{\mathbf{ir}}}$, or of $b_{\mathbf{ir}}, r_{\mathbf{ir}}, x_{\mathbf{i\hat{r}}}, x_{\mathbf{ir}}$ such that $\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = \frac{c_{\mathbf{ir}}}{g} = h^{r_{\mathbf{ir}}} \wedge d_{\mathbf{i\hat{r}}} = Y_{\mathbf{i\hat{r}}}^{x_{\mathbf{i\hat{r}}}} \wedge X_{\mathbf{i\hat{r}}} = g^{x_{\mathbf{i\hat{r}}}} \wedge v_{\mathbf{ir}} = Y_{\mathbf{ir}}^{x_{\mathbf{ir}}} \wedge X_{\mathbf{ir}} = g^{x_{\mathbf{ir}}}$. We denote this NIZK by

$$
\begin{aligned}
AV\{b_{\mathbf{ir}}, r_{\mathbf{ir}}, x_{\mathbf{ir}}, \bar{r}_{\mathbf{i\hat{r}}}, \bar{r}_{\mathbf{ir}}, x_{\mathbf{i\hat{r}}} \,|\, & (\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = c_{\mathbf{ir}} = h^{r_{\mathbf{ir}}} \wedge v_{\mathbf{ir}} = Y_{\mathbf{ir}}^{x_{\mathbf{ir}}} \wedge X_{\mathbf{ir}} = g^{x_{\mathbf{ir}}}) \vee \\
& (\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = \frac{c_{\mathbf{ir}}}{g} = h^{r_{\mathbf{ir}}} \wedge d_{\mathbf{i\hat{r}}} = g^{\bar{r}_{\mathbf{i\hat{r}}}} \wedge v_{\mathbf{ir}} = g^{\bar{r}_{\mathbf{ir}}}) \vee \\
& (\frac{c_{\mathbf{ir}}}{g^{b_{\mathbf{ir}}}} = \frac{c_{\mathbf{ir}}}{g} = h^{r_{\mathbf{ir}}} \wedge \ d_{\mathbf{i\hat{r}}} = Y_{\mathbf{i\hat{r}}}^{x_{\mathbf{i\hat{r}}}} \wedge X_{\mathbf{i\hat{r}}} = g^{x_{\mathbf{i\hat{r}}}} \wedge \\
& v_{\mathbf{ir}} = Y_{\mathbf{ir}}^{x_{\mathbf{ir}}} \wedge X_{\mathbf{ir}} = g^{x_{\mathbf{ir}}})\}
\end{aligned}
$$

Following the approach proposed by Camenisch and Stadler [22] we construct this NIZK as follows:

$$\tilde{F}_1 = DL(h, c_{\mathbf{ir}}) \otimes \Big[DL(Y_{\mathbf{ir}}, v_{\mathbf{ir}}) \cap DL(g, X_{\mathbf{ir}})\Big]$$

$$\tilde{F}_2 = DL(h, c_{\mathbf{ir}}/g) \otimes DL(g, d_{\mathbf{i\hat{r}}}) \otimes DL(g, v_{\mathbf{ir}})$$

$$\tilde{F}_3 = DL(h, c_{\mathbf{ir}}/g) \otimes \Big[DL(Y_{\mathbf{i\hat{r}}}, d_{\mathbf{i\hat{r}}}) \cap DL(g, X_{\mathbf{i\hat{r}}})\Big] \otimes \Big[DL(Y_{\mathbf{ir}}, v_{\mathbf{ir}}) \cap DL(g, X_{\mathbf{ir}})\Big]$$

Therefore we need to show

$$\tilde{F} = \tilde{F}_1 \cup \tilde{F}_2 \cup \tilde{F}_3$$

To prove the knowledge of either $\tilde{F}_1$ or $\tilde{F}_2$ or $\tilde{F}_3$, assuming that $\tilde{F}_\alpha$ is known, party $\mathbf{i}$ proceeds as follows:

1. Choose $\bar{V} = (\bar{v}_1, \bar{v}_2, \bar{v}_3, \bar{v}_4, \bar{v}_5, \bar{v}_6, \bar{v}_7, \bar{v}_8)$ with $\bar{v}_i \xleftarrow{\$} \mathbb{Z}_q$ and $\bar{w} = (\bar{w}_1, \bar{w}_2, \bar{w}_3)$ with $\bar{w}_\alpha = 0$ and $\bar{w}_i \xleftarrow{\$} \mathbb{Z}_q$ for $i \neq \alpha$, and compute $t_1 = c_{\mathbf{ir}}^{\bar{w}_1} h^{\bar{v}_1}$, $t_2 = v_{\mathbf{ir}}^{\bar{w}_1} Y_{\mathbf{ir}}^{\bar{v}_2}$, $t_3 = X_{\mathbf{ir}}^{\bar{w}_1} g^{\bar{v}_2}$, $t_4 = (\frac{c_{\mathbf{ir}}}{g})^{\bar{w}_2} h^{\bar{v}_3}$, $t_5 = d_{\mathbf{i\hat{r}}}^{\bar{w}_2} g^{\bar{v}_4}$, $t_6 = v_{\mathbf{ir}}^{\bar{w}_2} g^{\bar{v}_5}$, $t_7 = (\frac{c_{\mathbf{ir}}}{g})^{\bar{w}_3} h^{\bar{v}_6}$, $t_8 = d_{\mathbf{i\hat{r}}}^{\bar{w}_3} Y_{\mathbf{i\hat{r}}}^{\bar{v}_7}$, $t_9 = X_{\mathbf{i\hat{r}}}^{\bar{w}_3} g^{\bar{v}_7}$, $t_{10} = v_{\mathbf{ir}}^{\bar{w}_3} Y_{\mathbf{ir}}^{\bar{v}_8}$ and $t_{11} = X_{\mathbf{ir}}^{\bar{w}_3} g^{\bar{v}_8}$.
2. $H = \mathcal{H}(h, c_{\mathbf{ir}}, Y_{\mathbf{ir}}, v_{\mathbf{ir}}, g, X_{\mathbf{ir}}, \frac{c_{\mathbf{ir}}}{g}, d_{\mathbf{i\hat{r}}}, Y_{\mathbf{i\hat{r}}}, X_{\mathbf{i\hat{r}}}, t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11})$.
3. $\Gamma = (\gamma_1, \gamma_2, \gamma_3)$ where

$$\gamma_i = \begin{cases} H - (\bar{w}_1 + \bar{w}_2 + \bar{w}_3) \pmod q, & \textit{if} \quad i = \alpha \\ \bar{w}_i, & \textit{otherwise} \end{cases}$$

4. $R = (r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9, r_{10}, r_{11})$ where $r_1 = \bar{v}_1 - \gamma_\alpha x_1$, $r_2 = \bar{v}_2 - \gamma_\alpha x_2$, $r_3 = \bar{v}_2 - \gamma_\alpha x_2$, $r_4 = \bar{v}_3 - \gamma_\alpha x_3$, $r_5 = \bar{v}_4 - \gamma_\alpha x_4$, $r_6 = \bar{v}_5 - \gamma_\alpha x_5$, $r_7 = \bar{v}_6 - \gamma_\alpha x_6$, $r_8 = \bar{v}_7 - \gamma_\alpha x_7$, $r_9 = \bar{v}_7 - \gamma_\alpha x_7$, $r_{10} = \bar{v}_8 - \gamma_\alpha x_8$ and $r_{11} = \bar{v}_8 - \gamma_\alpha x_8$ (all equations are modulo q), in which $(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) = (r_{\mathbf{ir}}, x_{\mathbf{ir}}, 0, 0, 0, 0, 0, 0)$ if $\alpha = 1$, and $(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) = (0, 0, r_{\mathbf{ir}}, \bar{r}_{\widehat{\mathbf{ir}}}, \bar{r}_{\mathbf{ir}}, 0, 0, 0)$ if $\alpha = 2$, and $(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) = (0, 0, 0, 0, 0, 0, r_{\mathbf{ir}}, x_{\widehat{\mathbf{ir}}}, x_{\mathbf{ir}})$ if $\alpha = 3$.

The resulting proof is $(\gamma_1, \gamma_2, \gamma_3, r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9, r_{10}, r_{11})$. The validity of the proof can be checked by first re-constructing the commitments. That is,

$$
\begin{aligned}
t'_1 &= c_{\mathbf{ir}}^{\gamma_1} h^{r_1} & t'_2 &= v_{\mathbf{ir}}^{\gamma_1} Y^{r_2} & t'_3 &= X_{\mathbf{ir}}^{\gamma_1} g^{r_3} \\
t'_4 &= (\frac{c_{\mathbf{ir}}}{g})^{\gamma_2} h^{r_4} & t'_5 &= d_{\widehat{\mathbf{ir}}}^{\gamma_2} g^{r_5} & t_6 &= v_{\mathbf{ir}}^{\gamma_2} g^{r_6} \\
t'_7 &= (\frac{c_{\mathbf{ir}}}{g})^{\gamma_3} h^{r_7} & t'_8 &= d_{\widehat{\mathbf{ir}}}^{\gamma_3} Y_{\widehat{\mathbf{ir}}}^{r_8} & t'_9 &= X_{\widehat{\mathbf{ir}}}^{\gamma_3} g^{r_9} \\
t'_{10} &= v_{\mathbf{ir}}^{\gamma_3} Y_{\mathbf{ir}}^{r_{10}} & t'_{11} &= X_{\mathbf{ir}}^{\gamma_3} g^{r_{11}}
\end{aligned}
$$

and then checks the following condition

$$\gamma_1 + \gamma_2 + \gamma_3 = \mathcal{H}(h, c_{\mathbf{ir}}, Y_{\mathbf{ir}}, v_{\mathbf{ir}}, g, X_{\mathbf{ir}}, \frac{c_{\mathbf{ir}}}{g}, d_{\widehat{\mathbf{ir}}}, Y_{\widehat{\mathbf{ir}}}, X_{\widehat{\mathbf{ir}}}, t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11})$$

B Simplified UTXO model

Bitcoin [55] and other similar cryptocurrencies use the concept of *unspent transaction output*, or UTXO, that represents an indivisible amount of currency locked to an owner [16]. Each transaction contains a certain number of consumed UTXO, named transactions inputs, and created UTXO, named transaction outputs. In particular, a UTXO defines a number representing a certain amount of currency and a *locking script* specifying the conditions that has to be satisfied to use the UTXO as a transaction input (*i.e.* spend the UTXO). Note that each created UTXO can have a different recipient and that, in case the amount of currency that has to be transferred is smaller than the sum of the inputs, a *change* UTXO is created, *i.e.*, an UTXO that is still owned by the sender of the transaction. Miners have the role of checking if the unlocking conditions are satisfied and if the sum of the inputs is greater than the sum of the outputs.

In order to focus on the novel aspects of our protocol, we represent cryptocurrency transactions under a simplified version of the Bitcoin UTXO model and we only consider operations of the "Pay to Public Key" (P2PK) output type.

Representing Addresses: An address $Addr = pk$ is simply a signature verification key associated to a certain secret key sk, where pk and sk are generated by the key generation algorithm $\text{Gen}(k)$ and subsequent signatures σ are generated by the signature algorithm $\text{Sig}(sk, .) = sig_{sk}(.)$;

Representing Transactions: We represent a transaction in our simplified UTXO model by the tuple $\mathsf{tx} = (\mathsf{id}, \mathsf{In}, \mathsf{Out}, \mathsf{Sig})$, where $\mathtt{id} \in \{0,1\}^\kappa$ is a unique

transaction identification, $\mathsf{In} = \{(\mathtt{id}_1, \mathtt{in}_1), \ldots, (\mathtt{id}_m, \mathtt{in}_m)\}$ is a set of pairs of previous transaction id's $\mathtt{id} \in \{0,1\}^\kappa$ and their values $\mathtt{in} \in \mathbb{N}$, $\mathsf{Out} = \{(\mathtt{out}_1, Addr_1), \ldots, (\mathtt{out}_n, Addr_n)\}$ is a set of pairs of values $\mathtt{out} \in \mathbb{N}$ and addresses $Addr$ and $\mathsf{Sig} = \{\sigma_1, \ldots, \sigma_m\}$ is a set of signatures σ.

Transaction Validity: A transaction $\mathsf{tx} = (\mathtt{id}, \mathsf{In}, \mathsf{Out}, \mathsf{Sig})$ is considered valid if, for all $(\mathtt{id}_i, \mathtt{in}_i) \in \mathsf{In}$ and $(\mathtt{out}_j, Addr_j) \in \mathsf{Out}$, the following conditions hold:

1. There exists a valid transaction $\mathsf{tx}_i = (\mathtt{id}_i, \mathsf{In}_i, \mathsf{Out}_i, \mathsf{Sig}_i)$ in the public ledger such that $(\mathtt{in}_i, Addr_i) \in \mathsf{Out}_i$.
2. There exists $\sigma_i \in \mathsf{Sig}$ such that σ_i is a valid signature of $\mathtt{id}|\mathsf{In}|\mathsf{Out}$ under $Addr_i$, *i.e.*, $Ver(pk_i, \mathtt{id}|\mathsf{In}|\mathsf{Out}, \sigma_i) = True$.
3. It holds that $\sum_{i=1}^{m} \mathtt{in}_i = \sum_{j=1}^{n} \mathtt{out}_j$.

Generating Transactions: A party controlling the corresponding signing keys $sk_1, \ldots, sk_m$ for valid UTXO addresses $Addr_1, \ldots, Addr_m$ containing values $\mathtt{in}_1$, $\ldots, \mathtt{in}_m$ can generate a transaction that transfers the funds in these addresses to output addresses $Addr_{out,1}, \ldots, Addr_{out,n}$ by proceeding as follows:

1. Choose a unique $\mathtt{id} \in \{0,1\}^\kappa$.
2. Choose values $\mathtt{out}_1, \ldots, \mathtt{out}_n$ such that $\sum_{i=1}^{m} \mathtt{in}_i = \sum_{j=1}^{n} \mathtt{out}_j$.
3. Generate sets $\mathtt{in}$ and Out as described above and sign $\mathtt{id}|\mathsf{In}|\mathsf{Out}$ with the signing keys corresponding to $Addr_1, \ldots, Addr_m$, *i.e.*, $\sigma_i = sig_{sk_i}(\mathtt{id}|\mathsf{In}|\mathsf{Out})$ for $i = 1, \cdots, m$, obtaining $\mathsf{Sig} = \{\sigma_1, \ldots, \sigma_m\}$.
4. Output $\mathsf{tx} = (\mathtt{id}, \mathsf{In}, \mathsf{Out}, \mathsf{Sig})$.

C Publicly Verifiable Secret Sharing (PVSS)

In this section, we present the PVSS protocol π_{PVSS} from [24] described in Figure 12. In order to instantiate this protocol, the NIZKs described below are also necessary.

NIZK for Discrete Logarithm Equality (DLEQ): This NIZK from [25] is used to prove that, given $g_1, ..., g_m$ and $x_1, ..., x_m$, the discrete logarithms of every x_i with base g_i are equal, *i.e.*, $x_i = g_i^\alpha$ for all $i = 1, \ldots, m$ for some $\alpha \in \mathbb{Z}_q$ (the same α for all i). It is denoted as $DLEQ((g_i, x_i)_I, (h_i, y_i)_i)$. In this NIZK, the prover computes $e = H(g_1, \ldots, g_m, x_1, \ldots, x_m, a_1, \ldots, a_m)$, for $H(\cdot)$ a random oracle (that will be instantiated by a cryptographic hash function) and z as above. The proof is $(a_1, \ldots, a_m, e, z)$. The verifier checks that $e = H(g_1, \ldots, g_m, x_1, \ldots, x_m, a_1, \ldots, a_m)$ and that $a_i = g_i^z x_i^e$ for all i.

NIZK for Low-Degree Exponent Interpolation (*LDEI*): This NIZK from [24] is used to prove that, given generators $g_1, g_2, \ldots, g_m$ of a cyclic group $\mathbb{G}_q$ of prime order q, pairwise distinct elements $\alpha_1, \alpha_2, \ldots, \alpha_m$ in $\mathbb{Z}_q$ and an integer $1 \leq k < m$ known by a prover and a verifier, for $p(x)$ known by the prover, it holds that $(x_1, x_2, \ldots, x_m) \in \left\{ \left(g_1^{p(\alpha_1)}, g_2^{p(\alpha_2)}, \ldots, g_m^{p(\alpha_m)} \right) : p \in \mathbb{Z}_q[X], \deg p \leq k \right\}$.

We denote this NIZK by $LDEI((g_i)_{i\in[m]}, (\alpha_i)_{i\in[m]}, k, (x_i)_{i\in[m]})$. In this NIZK, the sender chooses $r \in \mathbb{Z}_q[X]_{\leq k}$ uniformly at random and computes $a_i = g_i^{r(\alpha_i)}$ for all $i = 1, \ldots, m$, $e = H(x_1, x_2, \ldots, x_m, a_1, a_2, \ldots, a_m)$ and $z = e \cdot p + r$. The proof is then $(a_1, a_2, \ldots, a_m, e, z)$. The verifier checks that $z \in \mathbb{Z}_q[X]_{\leq k}$, that $x_i^e \cdot a_i = g_i^{z(\alpha_i)}$ holds for all $i = 1, \ldots, m$ and that $e = H(x_1, x_2, \ldots, x_m, a_1, a_2, \ldots, a_m)$.

Protocol π_{PVSS} from [24]

Let h be a generator of a group $\mathbb{G}_q$ of order q. Let $H(\cdot)$ be a random oracle. Protocol π_{PVSS} is run between n parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$, a dealer D and an external verifier V (in fact any number of external verifiers) who have access to a public ledger where they can post information for later verification.

1. **Setup:** Party $\mathcal{P}_i$ generates a secret key $sk_i \leftarrow \mathbb{Z}_q$, a public key $pk_i = h^{sk_i}$ and registers the public key pk_i by posting it to the public ledger, for $1 \leq i \leq n$.
2. **Distribution:** The dealer D samples a polynomial $p(X) \leftarrow \mathbb{Z}_q[X]_{\leq t+\ell-1}$ (where $t = \lceil \frac{n}{2} \rceil - 1$ and ℓ is the number of secrets) and sets $s_0 = p(0), s_1 = p(-1), \ldots, s_{\ell-1} = p(-(\ell-1))$. The secrets are defined to be $S_0 = h^{s_0}, S_1 = h^{s_1}, \ldots, S_{\ell-1} = h^{s_{\ell-1}}$. D computes Shamir shares $\sigma_i = p(i)$ for $1 \leq i \leq n$. D encrypts the shares as $\hat{\sigma}_i = pk_i^{\sigma_i}$ and publishes $(\hat{\sigma}_1, \ldots, \hat{\sigma}_n)$ in the public ledger along with the proof $LDEI$ that $\hat{\sigma}_i = pk_i^{p(i)}$ for some p of degree at most $t+\ell-1$.
3. **Verification:** The verifier checks the proof $LDEI$.
4. **Share decryption**[a] **(for $\mathcal{P}_i$):** On input $\hat{\sigma}_i, pk_i$, decrypt share $\tilde{\sigma}_i = \hat{\sigma}_i^{\frac{1}{sk_i}} = h^{\sigma_i}$ and publish it in the ledger together with $PROOF_i = DLEQ((h, \tilde{\sigma}_i), (pk_i, \hat{\sigma}_i))$ (showing that the decrypted share $\tilde{\sigma}_i$ corresponds to $\hat{\sigma}_i$).
5. **Share decryption verification:** Apply the verification algorithm of the DLEQ proof $PROOF_i$ and complain if this is not correct.
6. **Secret reconstruction algorithm** $Rec_{\mathcal{Q}}$**:** On input $\{\tilde{\sigma}_i\}_{i\in\mathcal{Q}}$ for a set $\mathcal{Q}$ of exactly $n-t$ indices, for $j \in [\ell-1]$, Set $\lambda_i^{(j)} = \prod_{m:m\in\mathcal{Q},m\neq i} \frac{-j-m}{i-m}$ for all $i \in \mathcal{Q}$ and compute $S_j = \prod_{i\in\mathcal{Q}} (\tilde{\sigma}_i)^{\lambda_i^{(j)}} = \prod_{i\in\mathcal{Q}} h^{p(i)\lambda_i^{(j)}} = h^{p(-j)} = h^{s_j}$, and publish the values S_j.

[a] Amortized share decryption (for $\mathcal{P}_i$): If the PVSS has been used several times where $\mathcal{P}_i$ has received in each case a share $\hat{\sigma}_i^a$, $\mathcal{P}_i$ can decrypt shares as above but publish one single proof $PROOF_i = DLEQ((h, (\tilde{\sigma}_i^a)_a), (pk_i, (\hat{\sigma}_i^a)_a))$.

Fig. 12: Protocol π_{PVSS} from [24]

Definition 1 (Definition 4 from [24]). ***Indistinguishability of secrets (IND1-secrecy)*** *We say that the PVSS is IND1-secret if for any polynomial time adversary $\mathcal{A}$ corrupting at most $t-1$ parties, $\mathcal{A}$ has negligible advantage in the following game played against a challenger.*

1. *The challenger runs the Setup phase of the PVSS as the dealer and sends all public information to $\mathcal{A}$. Moreover, it creates secret and public keys for all honest parties, and sends the corresponding public keys to $\mathcal{A}$.*

2. *$\mathcal{A}$ creates secret keys for the corrupted parties and sends the corresponding public keys to the challenger.*
3. *The challenger chooses values $\mathbf{x}_0$ and $\mathbf{x}_1$ at random in the space of secrets. Furthermore it chooses $b \leftarrow \{0,1\}$ uniformly at random. It runs the Distribution phase of the protocol with $\mathbf{x}_0$ as secret. It sends $\mathcal{A}$ all public information generated in that phase, together with $\mathbf{x}_b$.*
4. *$\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.*

The advantage of $\mathcal{A}$ is defined as $|\Pr[b = b'] - 1/2|$.

Proposition 1 (Proposition 3 from [24]). *Protocol π_{PVSS} is IND1-secret under the DDH assumption.*

D Proof of Theorem 1

In this section, we provide a full proof of security for Protocol Π_{FPA}. In order to prove Theorem 1, we first the following auxiliary Lemmas:

Lemma 1. *Under the DDH Assumption, the Pedersen commitment scheme [58] is computationally binding and unconditionally hiding:*

- *Computationally binding: under the DL assumption, for any PPT algorithm the probability $\epsilon(q)$ of finding $s_1, t_1, s_2, t_2 \in \mathbb{Z}_q$ such that $s_1 \neq s_2$ and $com(s_1, t_2) = com(s_2, t_2)$ is $\mathsf{negl}(q)$.*
- *Unconditionally hiding: for any $s_1, s_2 \in \mathbb{Z}_q$ and $t_1, t_2 \xleftarrow{\$} \mathbb{Z}_q$, it holds that $|\Pr[\mathcal{D}(com(s_1, t_1)) = 1] - \Pr[\mathcal{D}(com(s_2, t_2)) = 1]| = \mathsf{negl}(q)$ for any distiguisher $\mathcal{D}$, i.e., $\{com(s_1, t_2)\}_{s_1 \in \mathbb{Z}_q, t_1 \xleftarrow{\$} \mathbb{Z}_q}$ and $\{com(s_1, t_2)\}_{s_2 \in \mathbb{Z}_q, t_2 \xleftarrow{\$} \mathbb{Z}_q}$ are statistically indistiguishable.*

Proof. It is proven in [58] that the Pedersen commitment scheme is computationally binding and unconditionally hiding under the assumption that the Discrete Logarithm problem is hard, which is implied by the DDH assumption.

Lemma 2. *Under the DDH Assumption and in the random oracle model, there exists a EUF-CMA [38] secure digital signature scheme.*

Proof. There exist a number of digital signature schemes whose security is implied by the DDH assumption in the random oracle model, *e.g.*, [59] and [13].

Lemma 3 (Theorem 5 from [21]). *Under the DDH Assumption and in the random oracle model, the range proofs computed in Stage 1 of Π_{FPA} guarantee the zero knowledge, the proof of knowledge and the soundness properties.*

Lemma 4 (From [22]). *Under the DDH Assumption and in the random oracle model, the NIZKs BV , AV, NW and CC computed respectively in Stage 2, Stage 3 and the Recovery Stage of Π_{FPA} have the zero knowledge, the proof of knowledge and the soundness properties.*

Lemma 5 (Lemma 2 from [4]). *Under the DDH Assumption, given* $X_{\boldsymbol{ir}} = g^{x_{\boldsymbol{ir}}}$ *with* $x_{\boldsymbol{ir}} \xleftarrow{\$} \mathbb{Z}_q$ *and* $\boldsymbol{i} \in [1,n]$, $Y_{\boldsymbol{ir}} = \prod_{k=1}^{i-1} g^{x_{kr}} / \prod_{k=\boldsymbol{i}+1}^{n} g^{x_{kr}} = g^{(\sum_{k=1}^{i-1} x_{kr} - \sum_{k=i+1}^{n} x_{kr})}$ *and* $y_{\boldsymbol{ir}} = \sum_{k=1}^{\boldsymbol{i}-1} x_{k\boldsymbol{r}} - \sum_{k=\boldsymbol{i}+1}^{n} x_{k\boldsymbol{r}}$ *with* $\boldsymbol{i} \in [1,n]$, $\boldsymbol{i}, \boldsymbol{i}' \in [1,n]$ *such that* $\boldsymbol{i} \neq \boldsymbol{i}'$, $g^{\bar{r}_{\boldsymbol{ir}}}$ *with* $\bar{r}_{\boldsymbol{ir}} \xleftarrow{\$} \mathbb{Z}_q$ *and* $\boldsymbol{i} \in [1,n] \setminus \{\boldsymbol{i}, \boldsymbol{i}'\}$, $g^{\bar{r}_{i'r}}$, $g^{\bar{r}_{i''r}}$, $\Phi \subseteq \{x_{\boldsymbol{ir}} : \boldsymbol{i} \in [1,n] \setminus \{\boldsymbol{i}, \boldsymbol{i}'\}\}$ *and a challenge* $\Omega \in \{A, B\}$, *it is computationally hard to find if* $\Omega = A$ *or* $\Omega = B$, *where:*

$$A = (g, \Phi, g^{x_{1r}z_{1r}}, g^{x_{2r}z_{2r}}, \ldots, g^{x_{i'-1r}z_{i'-1r}}, g^{x_{i'r}\bar{r}_{i'r}}, g^{x_{i'+1r}z_{i'+1r}}, \ldots, g^{x_{i''r}y_{i''r}}, \ldots, g^{x_{nr}z_{nr}})$$

$$B = (g, \Phi, g^{x_{1r}z_{1r}}, g^{x_{2r}z_{2r}}, \ldots, g^{x_{i'-1r}z_{i'-1r}}, g^{x_{i'r}\bar{r}_{i'r}}, g^{x_{i'+1r}z_{i'+1r}}, \ldots, g^{x_{i''r}\bar{r}_{i''r}}, \ldots, g^{x_{nr}z_{nr}})$$

where $z_{\boldsymbol{ir}}$ *is either* $y_{\boldsymbol{ir}}$ *(note that when* $z_{\boldsymbol{ir}} = y_{\boldsymbol{ir}}$ *the value* $g^{x_{ir}z_{ir}}$ *is equal to the message* $v_{\boldsymbol{ir}} = Y_{\boldsymbol{ir}}^{x_{\boldsymbol{ir}}} = g^{x_{ir}y_{ir}}$ *of* Π_{FPA} *representing a no veto) or* $\bar{r}_{\boldsymbol{ir}}$ *(note that when* $z_{\boldsymbol{ir}} = \bar{r}_{\boldsymbol{ir}}$, *where* $\bar{r}_{\boldsymbol{ir}} \xleftarrow{\$} \mathbb{Z}_q$, *the value* $g^{x_{ir}z_{ir}}$ *is indistinguishable from the message* $v_{\boldsymbol{ir}} = g^{\bar{r}_{ir}}$ *of* Π_{FPA} *representing a veto) for* $\boldsymbol{i} \in [1,n] \setminus \{\boldsymbol{i}, \boldsymbol{i}'\}$, *and* Φ *is chosen by an adversary* $\mathcal{A}$. *Intuitively, it is not possible to distinguish two executions in which there is at least one veto but the number of parties vetoing is different and it is not possible to learn if a party vetoed or not by checking* $v_{\boldsymbol{ir}}$.

Lemma 6 (Lemma 3 from [4]). *Under the DDH Assumption, let* $\mathcal{H}$ *be a set of honest parties and* $\mathcal{C}$ *be a set of parties corrupted by an adversary* $\mathcal{A}$. *For each* $\mathcal{P}_h \in \mathcal{H}$, *let* $v_{h\boldsymbol{r}}$ *be her message in Stages 2 or 3 during a round* $\boldsymbol{r}$ *of* Π_{FPA}, *corresponding to an input bit* $b_{h\boldsymbol{r}}$. *Then,* $\mathcal{A}$ *learns no more than* $\bigvee_{\mathcal{P}_h \in \mathcal{H}} b_{h\boldsymbol{r}}$. *In particular, in case* $\bigvee_{\mathcal{P}_h \in \mathcal{H}} b_{h\boldsymbol{r}} = 0$, $\mathcal{A}$ *learns that* $b_{h\boldsymbol{r}} = 0$ *for each* $\mathcal{P}_h \in \mathcal{H}$. *On the other hand, in case* $\bigvee_{\mathcal{P}_h \in \mathcal{H}} b_{h\boldsymbol{r}} = 1$*:*

- $\mathcal{A}$ *is not able to distinguish two executions in which the number of honest parties* $\mathcal{P}_h \in \mathcal{H}$ *with* $b_{h\boldsymbol{r}} = 1$ *is different.*
- *Let* $\mathcal{P}_{h_1}, \mathcal{P}_{h_2} \in \mathcal{H}$ *be honest parties with input bits* $b_{h_1\boldsymbol{r}}, b_{h_2\boldsymbol{r}}$ *such that* $b_{h_1\boldsymbol{r}} \neq b_{h_2\boldsymbol{r}}$ *and messages* $v_{h_1\boldsymbol{r}}, v_{h_2\boldsymbol{r}}$ *respectively. Then,* $\mathcal{A}$ *is not able to distinguish* $v_{h_1\boldsymbol{r}}$ *and* $v_{h_2\boldsymbol{r}}$.

Based on the above Lemmas, we prove Theorem 1, which we reproduce below for the sake of clarity.

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* $\mathcal{C}_i \in \mathcal{C}$.

Proof. In order to prove this theorem, we construct a simulator $\mathcal{S}_{FPA}$ (Figures 13, 14, 15 and 16) that performs an ideal execution with $\mathcal{F}_{FPA}$ and interacts with an internal copy of the adversary $\mathcal{A}$, simulates honest parties, $\mathcal{F}_{\mathsf{SC}}$ and the random oracle in an execution of Protocol Π_{FPA} with $\mathcal{A}$ in such a way that this execution is indistinguishable from an execution between $\mathcal{A}$ and an honest party in the real world. Throughout this execution, $\mathcal{S}_{FPA}$ perfectly emulates $\mathcal{F}_{\mathsf{SC}}$ and the random oracle unless stated otherwise. In order to show that an ideal execution with $\mathcal{S}_{FPA}$ and $\mathcal{F}_{FPA}$ is indistinguishable from a real execution of Π_{FPA} with $\mathcal{A}$ and honest parties, we argue that the view of $\mathcal{A}$ in the real world and of $\mathcal{S}_{FPA}$'s internal copy of $\mathcal{A}$ is indistinguishable. In particular:

Simulator $\mathcal{S}_{FPA}$ (Stage 1)

Let $\mathcal{H}$ be the set of simulated honest parties, $\mathcal{C}$ be the set of parties corrupted by the adversary $\mathcal{A}$, $\mathcal{C}_{\mathcal{H}}$ be the set of simulated members of the committee and $\mathcal{C}_{\mathcal{C}}$ be the set of members of the committee corrupted by the adversary $\mathcal{A}$.

Stage 1 - Setup.

- **Generating Parameters with Trapdoor:** $\mathcal{S}_{FPA}$ samples $g \xleftarrow{\$} \mathbb{G}$, $t \xleftarrow{\$} \mathbb{Z}_q$ and computes $h = g^t$. Trapdoor t will allow $\mathcal{S}_{FPA}$ to equivocate commitments later. When $\mathcal{A}$ queries $\mathcal{F}_{\mathsf{SC}}$ with (PARAM, sid), $\mathcal{S}_{FPA}$ returns these values (PARAM, sid, g, h).
- **Simulating Honest Parties:** $\mathcal{S}_{FPA}$ simulates each honest party $\mathcal{P}_h \in \mathcal{H}$ using dummy input bids $b'_h = b'_{h1}|\dots|b'_{hl}$ such that $b'_{h\mathbf{r}} \xleftarrow{\$} \{0,1\}$ for $\mathbf{r} = 1,\dots,l$. $\mathcal{S}_{FPA}$ follows the steps of an honest party with input b'_h in Π_{FPA} to do the following:
  1. Generate pk_h, transaction tx_h, shares $(\hat{\sigma}_{h1},\dots,\hat{\sigma}_{hm}, LDEI_h)$ of $g^{b'_h}$ and $h^{r_{b'_h}}$, proof of consistency $CC_h \leftarrow CC((pk_{\mathcal{C}_j})_{j\in[m]}, (j)_{j\in[m]}, g, h, c_h, (\hat{\sigma}_j)_{j\in[m]})$, then send (SETUP, $sid, \mathcal{P}_h, \mathsf{tx}_h, pk_h, (\hat{\sigma}_{h1},\dots,\hat{\sigma}_{hm}), LDEI_h, CC_h$) to each $\mathcal{C}_j \in \mathcal{C}_{\mathcal{C}}$.
  2. Upon receiving (SETUP-VERIFICATION, $sid, Sig_{\mathcal{C}_j,h}$) from all $\mathcal{C}_j \in \mathcal{C}_{\mathcal{C}}$, compute $SH1_h = \mathcal{H}(\mathsf{tx}_h, pk_h)$ and $SH2_h = \mathcal{H}((\hat{\sigma}_{h1},\dots,\hat{\sigma}_{hm}), LDEI_h, CC_h)$ and send (SETUP, $sid, \mathcal{P}_{\mathbf{i}}, \mathsf{tx}_h, pk_h, SH1_h, SH2_h, Sig_{\mathcal{C}_1,h},\dots,Sig_{\mathcal{C}_m,h}$)) to $\mathcal{F}_{\mathsf{SC}}$.
  3. Generate values $c_{h1},\cdots,c_{hl}$, $X_{h1},\cdots,X_{hl}$ (sample $x_{h\mathbf{r}} \xleftarrow{\$} \mathbb{Z}_q$, then $X_{h\mathbf{r}} = g^{x_{h\mathbf{r}}}$) and send them to all other parties.
  4. Upon receiving $X_{c1},\cdots,X_{cl}$ from $\mathcal{A}$ for all $\mathcal{P}_c \in \mathcal{C}$, computes Y_{hk} for each $\mathcal{P}_h \in \mathcal{H}$ and $k = 1,\dots,l$.
- **Handling messages from $\mathcal{A}$:**
  1. Upon receiving (SETUP, $sid, \mathcal{P}_a, \mathsf{tx}_a, pk_a, (\hat{\sigma}_{a1},\dots,\hat{\sigma}_{am}), LDEI_a, CC_a$) from $\mathcal{A}$ trough $\mathcal{P}_a \in \mathcal{C}$ to $\mathcal{C}_j \in \mathcal{C}_{\mathcal{H}}$, $\mathcal{S}_{FPA}$ follows the steps of $\Pi_{\mathcal{C}}$ to verify $\mathsf{tx}_a, (\hat{\sigma}_{a1},\dots,\hat{\sigma}_{am}), LDEI_a, CC_a$. If all the checks pass, computes the hashes $SH1_a = \mathcal{H}(\mathsf{tx}_a, pk_a)$ and $SH2_a = \mathcal{H}((\hat{\sigma}_{a1},\dots,\hat{\sigma}_{am}), LDEI_a, CC_a)$ and the signature $Sig_{\mathcal{C}_j,a} = sig_{sk'_{\mathcal{C}_j}}(SH1_a|SH2_a)$, then sends (SETUP-VERIFICATION, $sid, Sig_{\mathcal{C}_j,a}$) to $\mathcal{P}_a$.
  2. Upon receiving the message (SETUP, $sid, \mathcal{P}_a, \mathsf{tx}_a, pk_a, SH1_a|SH2_a$, $Sig_{\mathcal{C}_1,a},\dots,Sig_{\mathcal{C}_m,a}$) from $\mathcal{A}$ trough $\mathcal{P}_a \in \mathcal{C}$ to $\mathcal{F}_{\mathsf{SC}}$, $\mathcal{S}_{FPA}$ follows the steps of $\mathcal{F}_{\mathsf{SC}}$ to verify that $SH1_a = \mathcal{H}(\mathsf{tx}_a, pk_a)$ and $Sig_{\mathcal{C}_j,a}$ for each $\mathcal{C}_j \in \mathcal{C}_{\mathcal{H}} \cup \mathcal{C}_{\mathcal{C}}$, continuing the execution if the checks pass and simulating the recovery procedure otherwise.
  3. If the checks pass, $\mathcal{S}_{FPA}$ uses the knowledge extractor for the NIZKs in order to extract the bids b_a for each party $\mathcal{P}_a \in \mathcal{C}$ from their range proofs $(\pi_{b_a}, \pi_{change})$ included in the transactions tx_a. Then, $\mathcal{S}_{FPA}$ sends (BID, $sid, \mathsf{coins}(b_a + work)$) for each extracted bid b_a to $\mathcal{F}_{FPA}$ on behalf of $\mathcal{P}_a$.

Fig. 13: Simulator $\mathcal{S}_{FPA}$ (Stage 1).

- **Stage 1 - Setup:** $\mathcal{S}_{FPA}$ simulates each honest party $\mathcal{P}_h \in \mathcal{H}$ using a dummy input bid $b_h = b_{h1}|\dots|b_{hl}$ with $b_{h\mathbf{r}} \xleftarrow{\$} \{0,1\}$ for $\mathbf{r} = 1,\dots,l$ and computes the bit commitments $c_{h\mathbf{r}} = g^{b_{h\mathbf{r}}} h^{r_{h\mathbf{r}}}$ for $\mathbf{r} = 1,\dots,l$, the bid commitment $c_h = \prod_{\mathbf{r}=1}^{l} c_{h\mathbf{r}}^{2^{l-\mathbf{r}}} = g^{b_h} h^{\sum_{\mathbf{r}=1}^{l} 2^{l-\mathbf{r}} r_{h\mathbf{r}}}$, the range proofs $(\pi_{b_h}, \pi_{change})$, shares $(\hat{\sigma}_{h1},\dots,\hat{\sigma}_{hm}, LDEI_h)$ of $g^{b'_h}$ and $h^{r_{b'_h}}$ and the proof of consistency CC_h. However, by Lemma 1, due to the hiding property of the commitments, $c_{h\mathbf{r}}$ for $\mathbf{r} = 1,\dots,l$ and c_h are indistinguishable from the commitments of the

Simulator $\mathcal{S}_{FPA}$ (Stage 1 - Continuation)

Simulating an honest party vetoing or not vetoing. considering $\mathcal{P}_h \in \mathcal{H}$, in a round $\mathbf{r}$ during Stage 2 or Stage 3 requires $\mathcal{S}_{FPA}$ to compute $v_{h\mathbf{r}}$ as follows:

$$v_{h\mathbf{r}} = \begin{cases} Y_{h\mathbf{r}}^{x_{h\mathbf{r}}}, & \text{if } \mathcal{S}_{FPA} \text{ simulates a no veto} \\ \bar{r}_{h\mathbf{r}} \xleftarrow{\$} \mathbb{Z}_q, g^{\bar{r}_{h\mathbf{r}}}, & \text{if } \mathcal{S}_{FPA} \text{ simulates a veto} \end{cases}$$

and generate NIZKs showing that $v_{h\mathbf{r}}$ has been computed according to both the first $\mathbf{r}$ bits of the dummy input bid b'_h (*i.e.* $b'_{h1}, \ldots, b'_{h\mathbf{r}}$) and the outputs of the protocol in the previous rounds (*i.e.* $V_1, \ldots, V_{\mathbf{r}-1}$) by using the NIZKs simulators. Whether a simulated honest party vetoes or not is decided by $\mathcal{S}_{FPA}$ according to rules, defined in Stage 2 and Stage 3 of $\mathcal{S}_{FPA}$, meaning that the behavior of simulated honest parties is completely independent of the dummy input bid b'_h. In Stage 2, $\mathcal{S}_{FPA}$ uses the simulator of NIZK $BV\{b'_{h\mathbf{r}}, r_{h\mathbf{r}}, x_{h\mathbf{r}}, \bar{r}_{h\mathbf{r}} \mid (\frac{c_{h\mathbf{r}}}{g^{b'_{h\mathbf{r}}}} = c_{h\mathbf{r}} = h^{r_{h\mathbf{r}}} \wedge v_{h\mathbf{r}} = Y_{h\mathbf{r}}^{x_{h\mathbf{r}}} \wedge X_{h\mathbf{r}} = g^{x_{h\mathbf{r}}}) \vee (\frac{c_{h\mathbf{r}}}{g^{b'_{h\mathbf{r}}}} = \frac{c_{h\mathbf{r}}}{g} = h^{r_{h\mathbf{r}}} \wedge v_{h\mathbf{r}} = g^{\bar{r}_{h\mathbf{r}}})\}$ to generate a proof $BV_{h\mathbf{r}}$ and sends $(v_{h\mathbf{r}}, BV_{h\mathbf{r}})$ to all parties. In Stage 3, uses the generator of NIZK $AV\{b'_{h\mathbf{r}}, r_{h\mathbf{r}}, x_{h\mathbf{r}}, \bar{r}_{h\hat{\mathbf{r}}}, \bar{r}_{h\mathbf{r}}, x_{h\hat{\mathbf{r}}} \mid (\frac{c_{h\mathbf{r}}}{g^{b'_{h\mathbf{r}}}} = c_{h\mathbf{r}} = h^{r_{h\mathbf{r}}} \wedge v_{h\mathbf{r}} = Y_{h\mathbf{r}}^{x_{h\mathbf{r}}} \wedge X_{h\mathbf{r}} = g^{x_{h\mathbf{r}}}) \vee (\frac{c_{h\mathbf{r}}}{g^{b'_{h\mathbf{r}}}} = \frac{c_{h\mathbf{r}}}{g} = h^{r_{h\mathbf{r}}} \wedge d_{h\hat{\mathbf{r}}} = g^{\bar{r}_{h\hat{\mathbf{r}}}} \wedge v_{h\mathbf{r}} = g^{\bar{r}_{h\mathbf{r}}}) \vee (\frac{c_{h\mathbf{r}}}{g^{b'_{h\mathbf{r}}}} = \frac{c_{h\mathbf{r}}}{g} = h^{r_{h\mathbf{r}}} \wedge d_{h\hat{\mathbf{r}}} = Y_{h\hat{\mathbf{r}}}^{x_{h\hat{\mathbf{r}}}} \wedge X_{h\hat{\mathbf{r}}} = g^{x_{h\hat{\mathbf{r}}}} \wedge v_{h\mathbf{r}} = Y_{h\mathbf{r}}^{x_{h\mathbf{r}}} \wedge X_{h\mathbf{r}} = g^{x_{h\mathbf{r}}})\}$ to generate a proof $AV_{h\mathbf{r}}$ and sends $(v_{h\mathbf{r}}, AV_{h\mathbf{r}})$ to all parties.

Fig. 14: Simulator $\mathcal{S}_{FPA}$ (Stage 1 - Continuation).

corresponding parties in the real world. Moreover, by Lemma 3, due to the zero knowledge property of NIZKs, $(\pi_{b_h}, \pi_{change})$ are indistinguishable from the range proofs of the corresponding parties in the real world.
Similarly, by Lemma 4, due to the zero knowledge property of NIZKs, CC_h is indistinguishable from the NIZK of the corresponding party in the real world.
On the other hand, by Lemma 1, due to the binding property of the commitment, the bids b_c of each corrupted party $\mathcal{P}_a \in \mathcal{C}$, extracted by $\mathcal{S}_{FPA}$ using the NIZKs knowledge extractor (given the proof of knowledge property of NIZKs by Lemma 3), from the range proofs $(\pi_{b_a}, \pi_{change})$, cannot be changed later. Then, by Lemma 3, due to the soundness property of the NIZKs, it is computationally hard for the adversary $\mathcal{A}$ controlling each $\mathcal{P}_a \in \mathcal{C}$ to compute the range proofs while the bids are not in the expected range. Moreover, by Lemma 4, due to the soundness property of the NIZKs, it is ensured that at every round the inputs of all $\mathcal{P}_a \in \mathcal{C}$ are computed according to the initial bid b_c and the protocol rules.
Moreover, by Proposition 1, $(\hat{\sigma}_{h1}, \ldots, \hat{\sigma}_{hm}, LDEI_h)$ does not leak any information about the bids and it is guaranteed that the shares distribution is valid.
Finally, by adopting a EUF-CMA secure digital signature, whose existence is given by Lemma 2, in each round $\mathbf{r}$ no $\mathcal{P}_c \in \mathcal{C}$ has the chance to forge a signature $sig_{sk_\mathbf{i}}(msg_{\mathbf{r},c})$ for a certain message $msg_{\mathbf{r},c}$ pretending to be an honest party $\mathcal{P}_i$ in the real world, *i.e.*, no transaction tx_i nor a proof of cheating $\pi_c = (msg_{\mathbf{ri}}, sig_{sk_\mathbf{i}}(msg_{\mathbf{ri}}), msg'_{\mathbf{ri}})$, indicating that $\mathcal{P}_\mathbf{i}$ sent con-

Simulator $\mathcal{S}_{FPA}$ (Stages 2 and 3)

- **Stage 2 - Before First Veto.** Let $\mathbf{r}_v$ be the first round in which a veto occurs, *i.e.*, $V_{\mathbf{r}_v} \neq 1$. Then, for $\mathbf{r} = 1, \ldots, \mathbf{r}_v$ $\mathcal{S}_{FPA}$ waits for (ROUND-WINNER, $sid, b_{w\mathbf{r}}$) and proceeds as follows:
  1. *(honest winner)* If $b_{w\mathbf{r}} = 1$ and $b_{a\mathbf{r}} = 0$ for each $\mathcal{P}_a \in \mathcal{C}$ (note that $\mathcal{S}_{FPA}$ has previously extracted the bids of the corrupted parties), then $\mathcal{P}_w \notin \mathcal{C}$, *i.e.*, the winner is known to be honest by $\mathcal{S}_{FPA}$ but its identity will be known during Stage 4 only. In this case, $\mathcal{S}_{FPA}$ simulates one honest party $\mathcal{P}_{h_w} \in \mathcal{H}$ picked at random vetoing and simulates all the other honest parties not vetoing again, then goes to Stage 3.
  2. *(corrupted winner)* If $b_{w\mathbf{r}} = 1$ and $\mathcal{S}_{FPA}$ receives (LEAK-TO-WINNER, $sid, \mathbf{r}_w$) from $\mathcal{F}_{FPA}$, then $\mathcal{P}_w \in \mathcal{C}$ and its identity is known by $\mathcal{S}_{FPA}$. In this case, $\mathcal{S}_{FPA}$ simulates all honest parties $\mathcal{P}_h \in \mathcal{H}$ not vetoing and goes to Stage 3.
  3. *(know nothing)* If neither condition 1 nor 2 occurs, then nothing is known yet regarding $\mathcal{P}_w$ (neither its identity nor if $\mathcal{P}_w$ is corrupted or not). If $b_{w\mathbf{r}} = 1$ then $\mathcal{S}_{FPA}$ simulates all honest parties $\mathcal{P}_h \in \mathcal{H}$ vetoing and goes to Stage 3. If $b_{w\mathbf{r}} = 0$ then $\mathcal{S}_{FPA}$ simulates all honest parties $\mathcal{P}_h \in \mathcal{H}$ not vetoing and continues to simulate Stage 2.
  4. If an honest party $\mathcal{P}_h$ executing Π_{FPA} would initiate the Recovery Stage, $\mathcal{S}_{FPA}$ simulates $\mathcal{P}_h \in \mathcal{H}$ initiate it with the simulated $\mathcal{F}_{\mathsf{SC}}$. Then, at the end of each round $\mathbf{r}$, $\mathcal{S}_{FPA}$ receives (ABORT?, sid) from $\mathcal{F}_{FPA}$. If any $\mathcal{P}_a \in \mathcal{C}$ has been identified as a cheater in the Recovery Stage, $\mathcal{S}_{FPA}$ sends (ABORT, $sid, \mathcal{P}_a$) to $\mathcal{F}_{FPA}$, otherwise $\mathcal{S}_{FPA}$ sends (PROCEED, sid) to $\mathcal{F}_{FPA}$.
- **Stage 3 - After First Veto.** For $\mathbf{r} = \mathbf{r}_v, \ldots, l$ $\mathcal{S}_{FPA}$ waits for (ROUND-WINNER, $sid, b_{w\mathbf{r}}$) and proceeds as follows:
  1. *(honest winner)* If condition 1 of Stage 2 occurred or occurs during Stage 3, $\mathcal{S}_{FPA}$ eventually picks at random one simulated honest party $\mathcal{P}_{h_w} \in \mathcal{H}$ (in case it has not been already picked during Stage 2) and simulates her vetoing if $b_{wr} = 1$ and not vetoing if $b_{wr} = 0$ until $\mathbf{r} = l$ and simulates all the other honest parties not vetoing again
  2. *(corrupted winner)* If condition 2 of Stage 2 occurred or occurs during Stage 3, $\mathcal{S}_{FPA}$ simulates all honest parties $\mathcal{P}_h \in \mathcal{H}$ not vetoing until $\mathbf{r} = l$.
  3. *(know nothing)* If neither condition 1 nor 2 occurs, then nothing is known yet regarding $\mathcal{P}_w$ (neither its identity nor if $\mathcal{P}_w$ is corrupted or not).
     If $b_{w\mathbf{r}} = 1$ then $\mathcal{S}_{FPA}$ simulates all honest parties $\mathcal{P}_h \in \mathcal{H}$ vetoing. If $b_{w\mathbf{r}} = 0$ then $\mathcal{S}_{FPA}$ simulates all honest parties $\mathcal{P}_h \in \mathcal{H}$ not vetoing.
  4. If an honest party $\mathcal{P}_h$ executing protocol Π_{FPA} would initiate the Recovery Stage, $\mathcal{S}_{FPA}$ simulates $\mathcal{P}_h \in \mathcal{H}$ initiating it with the simulated $\mathcal{F}_{\mathsf{SC}}$. Then, at the end of each round $\mathbf{r}$, $\mathcal{S}_{FPA}$ receives (ABORT?, sid) from $\mathcal{F}_{FPA}$. If any $\mathcal{P}_a \in \mathcal{C}$ has been identified as a cheater in the Recovery Stage, $\mathcal{S}_{FPA}$ sends (ABORT, $sid, \mathcal{P}_a$) to $\mathcal{F}_{FPA}$, otherwise $\mathcal{S}_{FPA}$ sends (PROCEED, sid) to $\mathcal{F}_{FPA}$.

Fig. 15: Simulator $\mathcal{S}_{FPA}$ (Stages 2 and 3).

flicting messages, or $\pi_c = (msg_{\mathbf{ri}}, sig_{sk_\mathbf{i}}(msg_{\mathbf{ri}})$, indicating that $\mathcal{P}_\mathbf{i}$ sent an invalid message, can be forged. Additionally, $\mathcal{A}$ cannot deny having sent a message that it has signed in the past. Hence, proofs of cheating cannot be repudiated.

Simulator $\mathcal{S}_{FPA}$ (Stages 4 and Recovery)

- **Stage 4 - Output.** Once $\mathcal{S}_{FPA}$ receives (ANNOUNCE-WINNER-AND-REFUND, $sid, \mathcal{P}_w, b_w, \mathsf{coins}(b_\mathsf{i} + work)$) from $\mathcal{F}_{FPA}$:
  - If $\mathcal{P}_w \in \mathcal{C}$ and $\mathcal{P}_w$ deviates from the protocol during Stage 4, then $\mathcal{S}_{FPA}$ simulates the recovery procedure with the simulated $\mathcal{F}_\mathsf{SC}$. After $\mathcal{P}_w$ halts, $\mathcal{S}_{FPA}$ outputs whatever $\mathcal{P}_w$ outputs.
  - If $\mathcal{P}_w \in \mathcal{C}$ but $\mathcal{P}_w$ does not deviate from the protocol, then the execution is is concluded as in the real world.
  - If $\mathcal{P}_w \notin \mathcal{C}$ then the winning party $\mathcal{P}_w$ in the real world is one of the honest parties. In this case, $\mathcal{S}_{FPA}$ simulates $\mathcal{P}_w \in \mathcal{H}$ winning and opening her commitment c_w not to her dummy input bid b'_w but to the actual bid b_w received from $\mathcal{S}_{FPA}$. Note that $\mathcal{P}_w \in \mathcal{H}$ is, in general, different from the simulated winning honest party $\mathcal{P}_{h_w} \in \mathcal{H}$ that was picked at random during the simulation of Stage 2 or 3. $\mathcal{S}_{FPA}$ uses trapdoor t from Stage 1 - Setup to find a randomness r' such that $c_w = \mathrm{com}(b_w, r')$ by solving $b'_w + t \cdot r_w = b_w + t \cdot r'$ for r'.
- **Recovery Stage.** $\mathcal{S}_{FPA}$ perfectly emulates $\mathcal{F}_\mathsf{SC}$. In particular, in case $\mathcal{P}_w$ is corrupted and did not send (OUTPUT, $sid, \mathcal{P}_w, b_w, r_{b_w}, \{sig_{sk_k}(b_w)\}_{k\in[n]}$) to $\mathcal{F}_\mathsf{SC}$, $\mathcal{S}_{FPA}$ identifies $\mathcal{P}_w$ anyway by observing the extracted inputs bids b_a from each $\mathcal{P}_a \in \mathcal{C}$. Then, $\mathcal{S}_{FPA}$ simulates all honest parties $\mathcal{P}_h \in \mathcal{H}$ sending (RECOVERY-PAYMENT, sid, NW_h) to $\mathcal{F}_\mathsf{SC}$ where $NW_h \leftarrow NW\{(V_1 = 1 \wedge v_{h1} = Y_{h1}^{x_{h1}}) \vee \ldots \vee (V_l = 1 \wedge v_{h1} = Y_{k1}^{x_{h1}})\}$ is generated using the simulator for NW. Moreover, $\mathcal{S}_{FPA}$ simulates each committee member $\mathcal{C}_j \in \mathcal{C}_\mathcal{H}$ by following $\Pi_\mathcal{C}$. Indeed, upon receiving (OPEN, $sid, \mathcal{P}_a$), where $\mathcal{P}_a \in \mathcal{C}$, from $\mathcal{F}_\mathsf{SC}$, $\mathcal{S}_{FPA}$ simulates $\mathcal{C}_j$ using the share decryption procedure from π_{PVSS} on $\hat{\sigma}_{aj}$, obtaining $\tilde{\sigma}_{aj}, DLEQ_{aj}$. and sending (SHARE-DECRYPTION, $sid, (\hat{\sigma}_{a1}, \ldots, \hat{\sigma}_{am})$, $LDEI_a, CC_a, \tilde{\sigma}_{aj}, DLEQ_{aj}$)) to $\mathcal{F}_\mathsf{SC}$.

Fig. 16: Simulator $\mathcal{S}_{FPA}$ (Stages 4 and Recovery).

- **Stage 2 - Before First Veto and Stage 3 - After First Veto:** $\mathcal{S}_{FPA}$ simulates each honest party $\mathcal{P}_h \in \mathcal{H}$ vetoing or not vetoing in each round $\mathbf{r}$, by using the NIZKs simulator, in a way that is decided arbitrarily by $\mathcal{S}_{FPA}$ and that is completely independent of the dummy input bid b_h.

  In particular, $\mathcal{S}_{FPA}$ simulates each honest party $\mathcal{P}_h \in \mathcal{H}$ vetoing or not vetoing coherently with the extracted input bids b_c from each party $\mathcal{P}_a \in \mathcal{C}$ corrupted by the adversary $\mathcal{A}$ and the bit $b_{w\mathbf{r}}$ learnt from $\mathcal{F}_{FPA}$ in each round $\mathbf{r}$, *i.e.*, in case a corrupted party is known to be the winner, $\mathcal{S}_{FPA}$ simulates each honest party $\mathcal{P}_h \in \mathcal{H}$ not vetoing until $\mathbf{r} = l$. On the other hand, in case an honest party is known to be the winner, $\mathcal{S}_{FPA}$ simulates one honest party $\mathcal{P}_{h_w} \in \mathcal{H}$ picked at random vetoing or not vetoing according if $b_{w\mathbf{r}} = 1$ or $b_{w\mathbf{r}} = 0$ respectively and simulates all the other honest parties not vetoing again. Similarly, if nothing is known about the winner, $\mathcal{S}_{FPA}$ simulates all honest parties $\mathcal{P}_h \in \mathcal{H}$ vetoing or not vetoing according if $b_{w\mathbf{r}} = 1$ or $b_{w\mathbf{r}} = 0$ respectively. We will argue why the view of the adversary $\mathcal{A}$ in the simulation by $\mathcal{S}_{FPA}$ is indistinguishable from the real world execution.

By Lemma 4, due to the zero knowledge property of NIZKs, $BV_{h\mathbf{r}}$ and $AV_{h\mathbf{r}}$ are indistinguishable from the NIZKs of the corresponding parties in the real world. Moreover, by Lemma 5 and Lemma 6, it is proven that the inputs $v_{h\mathbf{r}}$ of the veto protocol and the output $V_{\mathbf{r}}$ of each round $\mathbf{r}$ are indistinguishable from the inputs of the corresponding parties and the output in the real world. On the other hand, $\mathcal{S}_{FPA}$ can compare the extracted bids b_a of each $\mathcal{P}_a \in \mathcal{C}$ with the output of each round $V_{\mathbf{r}}$ to discover if one of the honest parties in the real world in the winner of the auction. In that case, as described in the simulator, $\mathcal{S}_{FPA}$ simulates one honest party $\mathcal{P}_{h_w} \in \mathcal{H}$ picked at random behaving as winner. However, by Lemma 5 and Lemma 6, the adversary $\mathcal{A}$ cannot distinguish which honest party is the winner.

- **Stage 4 - Output:** in case one of the honest parties in the real world is the winner of the auction, in the output stage she will reveal her identity and open 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 $\mathcal{F}_{\mathsf{SC}}$. In the ideal world, as described in the simulator, $\mathcal{S}_{FPA}$ has to simulate $\mathcal{P}_w \in \mathcal{H}$ winning and opening her commitment c_w not to her dummy input bid b'_w but to the bid b_w of $\mathcal{P}_w$ in the real world, *i.e.*, equivocate the commitment. Note that $\mathcal{P}_w \in \mathcal{H}$ is, in general, different from the simulated winning honest party $\mathcal{P}_{h_w} \in \mathcal{H}$ that was picked at random during the simulation of Stage 2 or 3. However, by Lemma 5 and Lemma 6, the adversary $\mathcal{A}$ cannot distinguish if $\mathcal{P}_w$ is different from $\mathcal{P}_{h_w}$.
  Moreover, by Lemma 1, due to the unconditionally hiding property of the commitment, the adversary cannot learn that c_w initially was a commitment to b'_w instead of b_w.
- **Recovery:** $\mathcal{S}_{FPA}$ simulates aborts and corresponding recovery stages if $\mathcal{A}$ deviates from the protocol as in an actual execution of Π_{FPA}, *i.e.*, when an honest party would have triggered the Recovery Stage. Moreover, by Lemma 4, due to the zero knowledge property of NIZKs, NW_h for each $\mathcal{P}_h \in \mathcal{H}$ are indistinguishable corresponding NIZKs in the real world. Finally, by Proposition 1, it is guaranteed that the shares reconstruction is valid.

Hence, the view of $\mathcal{A}$ in the real world and of $\mathcal{S}_{FPA}$'s internal copy of $\mathcal{A}$ is indistinguishable, which concludes our proof.

E Proof of Theorem 2

In this section, we prove Theorem 2, which we reproduce below for the sake of clarity.

Theorem 2 *Under the DDH Assumption, Protocol Π_{SPA} securely computes $\mathcal{F}_{SPA}$ 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 $\mathcal{C}_i \in \mathcal{C}$.*

Proof. In order to prove this theorem, we construct a simulator $\mathcal{S}_{SPA}$ (Figure 17, Figure 18, Figure 19) that performs an ideal execution with $\mathcal{F}_{SPA}$ and interacts

Simulator $\mathcal{S}_{SPA}$ (Stages 1 and 2)

Let $\mathcal{H}$ be the set of simulated honest parties, $\mathcal{C}$ be the set of parties corrupted by the adversary $\mathcal{A}$, $\mathcal{C}_{\mathcal{H}}$ be the set of simulated members of the committee and $\mathcal{C}_{\mathcal{C}}$ be the set of members of the committee corrupted by the adversary $\mathcal{A}$.

- **Stage 1 - Setup.** Same as in $\mathcal{S}_{FPA}$.
  Simulating an honest party vetoing or not vetoing. Same as $\mathcal{S}_{FPA}$.
- **Stage 2 - Before First Veto.** For every round $\mathbf{r}$ before the first veto, $\mathcal{S}_{SPA}$ waits for (ROUND-WINNER, $sid, b_{w\mathbf{r}}$) and proceeds as follows:
  1. *(honest winner)* If $\mathcal{F}_{SPA}$ sends (ANNOUNCE-WINNER, $sid, \mathcal{P}_w$) to $\mathcal{S}_{SPA}$, then the winner is an honest party $\mathcal{P}_w$. Thus, $\mathcal{S}_{SPA}$ simulates $\mathcal{P}_w$ vetoing and proving to be the winner by sending to all the other parties (WINNER, $sid, \mathcal{P}_w, x_{w\mathbf{r}}$) and simulates all the other honest parties not vetoing again, then goes to Stage 3b.
  2. *(corrupted winner)* Upon receiving (LEAK-TO-WINNER, $sid, \mathbf{r}_w$) and subsequently (ANNOUNCE?, $sid, \mathcal{P}_w$) from $\mathcal{F}_{SPA}$, $\mathcal{S}_{SPA}$ learns the identity of the winner $\mathcal{P}_w$ and that is corrupted. If $\mathcal{A}$ sends a message (WINNER, $sid, \mathcal{P}_w, x_{w\mathbf{r}}$) then $\mathcal{S}_{SPA}$ replies (ANNOUNCE, sid) to $\mathcal{F}_{SPA}$, otherwise it replies (NOT-ANNOUNCE, sid) to $\mathcal{F}_{SPA}$. Then $\mathcal{S}_{SPA}$ simulates all honest parties $\mathcal{P}_h \in \mathcal{H}$ not vetoing and goes to Stage 3b.
  3. *(know nothing)* If neither condition 2 nor 3 occurs, then nothing is known yet regarding $\mathcal{P}_w$ (neither its identity nor if $\mathcal{P}_w$ is corrupted or not). If $b_{w\mathbf{r}} = 1$ then $\mathcal{S}_{SPA}$ simulates all honest parties $\mathcal{P}_h \in \mathcal{H}$ vetoing and goes to Stage 3a. If $b_{w\mathbf{r}} = 0$ then $\mathcal{S}_{SPA}$ simulates all honest parties $\mathcal{P}_h \in \mathcal{H}$ not vetoing and continues to simulate Stage 2.
  4. If an honest party $\mathcal{P}_h$ executing Π_{SPA} would initiate the Recovery Stage, $\mathcal{S}_{SPA}$ simulates $\mathcal{P}_h$ initiating it with the simulated $\mathcal{F}_{\mathsf{SC}}$. Then, at the end of each round $\mathbf{r}$, $\mathcal{S}_{SPA}$ receives (ABORT?, sid) from $\mathcal{F}_{SPA}$. If any $\mathcal{P}_a \in \mathcal{C}$ has been identified as a cheater in the Recovery Stage, $\mathcal{S}_{SPA}$ sends (ABORT, $sid, \mathcal{P}_a$) to $\mathcal{F}_{SPA}$, otherwise $\mathcal{S}_{SPA}$ sends (PROCEED, sid) to $\mathcal{F}_{SPA}$.

Fig. 17: Simulator $\mathcal{S}_{SPA}$ (Stages 1 and 2).

with an internal copy of the adversary $\mathcal{A}$, simulates honest parties, $\mathcal{F}_{\mathsf{SC}}$ and the random oracle in an execution of Protocol Π_{SPA} with $\mathcal{A}$ in such a way that this execution is indistinguishable from an execution between $\mathcal{A}$ and an honest party in the real world.

Throughout this execution, $\mathcal{S}_{SPA}$ perfectly emulates $\mathcal{F}_{\mathsf{SC}}$ and the random oracle unless stated otherwise. In order to show that an ideal execution with $\mathcal{S}_{SPA}$ and $\mathcal{F}_{SPA}$ is indistinguishable from a real execution of Π_{SPA} with $\mathcal{A}$ and honest parties, we argue that the view of $\mathcal{A}$ in the real world and of $\mathcal{S}_{SPA}$'s internal copy of $\mathcal{A}$ is indistinguishable. In particular:

- **Stage 1 - Setup:** Same as Theorem 1.
- **Stage 2 - Before First Veto and Stage 3a - After First Veto:** In terms of differences with respect of $\mathcal{S}_{FPA}$, in case an honest party is known to be the winner in a certain round $\mathbf{r}$, $\mathcal{S}_{SPA}$ immediately learns the identity of the winning party by receiving the message (ANNOUNCE-WINNER, $sid, \mathcal{P}_w$) from $\mathcal{F}_{SPA}$. Then $\mathcal{S}_{SPA}$ simulates $\mathcal{P}_w \in \mathcal{H}$ vetoing and proving to be the winner

Simulator $\mathcal{S}_{SPA}$ (Stage 3a and 3b)

- **Stage 3a - After First Veto** For every round $\mathbf{r}$ after the first veto, $\mathcal{S}_{SPA}$ waits for (ROUND-WINNER, $sid, b_{w\mathbf{r}}$) and proceeds as follows:
  1. *(honest winner)* If $\mathcal{F}_{SPA}$ sends (ANNOUNCE-WINNER, $sid, \mathcal{P}_w$) to $\mathcal{S}_{SPA}$, then the winner is an honest party $\mathcal{P}_w$. Thus, $\mathcal{S}_{SPA}$ simulates $\mathcal{P}_w \in \mathcal{H}$ vetoing and proving to be the winner by sending to all the other parties $x_{w\mathbf{r}}$, simulates all the other honest parties $\mathcal{H} \setminus \mathcal{P}_w$ not vetoing, then goes to Stage 3b.
  2. *(corrupted winner)* Upon receiving (LEAK-TO-WINNER, $sid, \mathbf{r}_w$) and subsequently (ANNOUNCE?, $sid, \mathcal{P}_w$) from $\mathcal{F}_{SPA}$, $\mathcal{S}_{SPA}$ learns the identity of the winner $\mathcal{P}_w$ and that is corrupted. If $\mathcal{A}$ sends a message (WINNER, $sid, \mathcal{P}_w, x_{w\mathbf{r}}$) then $\mathcal{S}_{SPA}$ replies (ANNOUNCE, sid) to $\mathcal{F}_{SPA}$, otherwise it replies (NOT-ANNOUNCE, sid) to $\mathcal{F}_{SPA}$. Then $\mathcal{S}_{SPA}$ simulates all honest parties $\mathcal{P}_h \in \mathcal{H}$ not vetoing and goes to Stage 3b.
  3. *(know nothing)* If neither condition 2 nor 3 occurs, then nothing is known yet regarding $\mathcal{P}_w$ (neither its identity nor if $\mathcal{P}_w$ is corrupted or not). If $b_{w\mathbf{r}} = 1$ then $\mathcal{S}_{SPA}$ simulates all honest parties $\mathcal{P}_h \in \mathcal{H}$ vetoing. If $b_{w\mathbf{r}} = 0$ then $\mathcal{S}_{SPA}$ simulates all honest parties $\mathcal{P}_h \in \mathcal{H}$ not vetoing.
  4. If an honest party $\mathcal{P}_h$ executing Π_{SPA} would initiate the Recovery Stage, $\mathcal{S}_{SPA}$ simulates $\mathcal{P}_h$ initiating it with the simulated $\mathcal{F}_{\mathsf{SC}}$. Then, at the end of each round $\mathbf{r}$, $\mathcal{S}_{SPA}$ receives (ABORT?, sid) from $\mathcal{F}_{SPA}$. If any $\mathcal{P}_a \in \mathcal{C}$ has been identified as a cheater in the Recovery Stage, $\mathcal{S}_{SPA}$ sends (ABORT, $sid, \mathcal{P}_a$) to $\mathcal{F}_{SPA}$, otherwise $\mathcal{S}_{SPA}$ sends (PROCEED, sid) to $\mathcal{F}_{SPA}$.
- **Stage 3b - After First Unique Veto.** If $\mathcal{P}_w$ is honest, $\mathcal{S}_{SPA}$ excludes it from Stage 3b. For all other rounds $\mathbf{r}$, $\mathcal{S}_{SPA}$ waits for (ROUND-WINNER, $sid, b_{w_2\mathbf{r}}$) and:
  1. *(honest second)* If $b_{w_2\mathbf{r}} = 1$ and $b_{a\mathbf{r}} = 0$ for each $\mathcal{P}_a \in \mathcal{C}$ (note that $\mathcal{S}_{SPA}$ has previously extracted the bids of the corrupted parties), then $\mathcal{S}_{SPA}$ learns that the second price bid was done by an honest party $\mathcal{P}_{w_2}$. In this case, $\mathcal{S}_{SPA}$ simulates one honest party $\mathcal{P}_{h_{w_2}}$ picked at random (excluding the 1st price winner $\mathcal{P}_w$ if it is honest) vetoing if $b_{w_2\mathbf{r}} = 1$ and not vetoing if $b_{w_2\mathbf{r}} = 0$ until $\mathbf{r} = l$ and simulates all the other honest parties not vetoing again.
  2. *(corrupted second)* If $b_{w_2\mathbf{r}} = 1$ and $\mathcal{S}_{SPA}$ receives (LEAK-TO-SECOND, $sid, \mathbf{r}_{w_2}$) from $\mathcal{F}_{SPA}$, then $\mathcal{S}_{SPA}$ learns that the second price bid belongs to a corrupted party $\mathcal{P}_{w_2} \in \mathcal{C}$. In this case, $\mathcal{S}_{SPA}$ simulates all honest parties $\mathcal{P}_h$ not vetoing until $\mathbf{r} = l$.
  3. *(know nothing)* If neither condition 2 nor 3 occurs, then nothing is known yet regarding $\mathcal{P}_{w_2}$ (neither its identity nor if $\mathcal{P}_{w_2}$ is corrupted or not). If $b_{w_2\mathbf{r}} = 1$ then $\mathcal{S}_{SPA}$ simulates all honest parties $\mathcal{P}_h \in \mathcal{H}$ vetoing. If $b_{w_2\mathbf{r}} = 0$ then $\mathcal{S}_{SPA}$ simulates all honest parties $\mathcal{P}_h \in \mathcal{H}$ not vetoing. Then continues to simulate Stage 3b until $\mathbf{r} = l$.
  4. If an honest party $\mathcal{P}_h$ executing Π_{SPA} would initiate the Recovery Stage, $\mathcal{S}_{SPA}$ simulates $\mathcal{P}_h \in \mathcal{H}$ initiating it with the simulated $\mathcal{F}_{\mathsf{SC}}$. Then, at the end of each round $\mathbf{r}$, $\mathcal{S}_{SPA}$ receives (ABORT?, sid) from $\mathcal{F}_{SPA}$. If any $\mathcal{P}_a \in \mathcal{C}$ has been identified as a cheater in the Recovery Stage, $\mathcal{S}_{SPA}$ sends (ABORT, $sid, \mathcal{P}_a$) to $\mathcal{F}_{SPA}$, otherwise $\mathcal{S}_{SPA}$ sends (PROCEED, sid) to $\mathcal{F}_{SPA}$.

Fig. 18: Simulator $\mathcal{S}_{SPA}$ (Stage 3a and 3b).

by sending to all the other parties (WINNER, $sid, \mathcal{P}_w, x_{w\mathbf{r}}$) and simulates all the other honest parties not vetoing again, then goes to Stage 3b. Notice

Simulator $\mathcal{S}_{SPA}$ (Stage 4 and Recovery)

- **Stage 4 - Output.** Once $\mathcal{S}_{SPA}$ receives $(\text{REFUND}, sid, b_{w_2}, \mathsf{coins}(b_{\mathtt{i}} + work))$ from $\mathcal{F}_{SPA}$:
  - If $\mathcal{P}_w$ is corrupted and deviates from the protocol during Stage 4, then $\mathcal{S}_{SPA}$ simulates the recovery procedure with the simulated $\mathcal{F}_{\mathsf{SC}}$.
  - If $\mathcal{P}_w$ is corrupted but does not deviate from the protocol, $\mathcal{S}_{SPA}$ follows the instructions of an honest party in Π_{SPA} to simulate honest parties.
  - If $\mathcal{P}_w$ is honest, $\mathcal{S}_{SPA}$ simulates $\mathcal{P}_w$ winning by creating a confidential transaction $\mathsf{tx}_{pay} = (\mathtt{id}_{pay}, \mathsf{In}_{pay}, \mathsf{Out}_{pay}, \mathsf{Sig}_{pay}, r_{b_w} - r_{change'_w}, \pi_{pay})$ where $\mathsf{In} = \{(\mathtt{id}, \text{com}(b_w, r_{b_w}))\}$, $\mathsf{Out} = \{(b_{w_2}, Addr_{auc}), (\text{com}(change'_w, r_{change'_w}), Addr_w)\}$, Sig_{pay} is left empty, $Addr_{auc}$ is the address of the auctioneer, $Addr_w$ is the address of $\mathcal{P}_w$ and π_{pay} is a NIZK showing that $change'_w$ is within $[0, 2^l - 1]$ (using the simulator for this NIZK). Finally, $\mathcal{S}_{SPA}$ simulates $\mathcal{P}_w$ sending $(\text{OUTPUT}, sid, \mathcal{P}_w, \mathsf{tx}_{pay}, \{sig_{sk_k}(\mathcal{P}_w | b_{w_2})\}_{k\in[n]})$ to $\mathcal{F}_{\mathsf{SC}}$.
  - After $\mathcal{P}_w$ halts, $\mathcal{S}_{SPA}$ outputs whatever $\mathcal{P}_w$ outputs.
- **Recovery Stage.** $\mathcal{S}_{SPA}$ perfectly emulates $\mathcal{F}_{\mathsf{SC}}$. In particular, in case $\mathcal{P}_w$ is corrupted and did not send $(\text{WINNER}, sid, \mathcal{P}_w, x_{w\mathbf{r}})$ to all the other parties, $\mathcal{S}_{SPA}$ knows her identity due to the message $(\text{ANNOUNCE?}, sid, \mathcal{P}_w)$ from $\mathcal{F}_{SPA}$. Then, $\mathcal{S}_{SPA}$ simulates all honest parties $\mathcal{P}_h \in \mathcal{H}$ sending $(\text{RECOVERY-DISHONEST-WINNER}, sid, NW_h)$ to $\mathcal{F}_{\mathsf{SC}}$ where $NW_h \leftarrow NW\{(V_1 = 1 \wedge v_{h1} = Y_{h1}^{x_{h1}}) \vee \ldots \vee (V_l = 1 \wedge v_{h1} = Y_{k1}^{x_{h1}})\}$ is generated using the simulator of NW. Moreover, $\mathcal{S}_{SPA}$ simulates each committee member $\mathcal{C}_j \in \mathcal{C}_{\mathcal{H}}$ by following $\Pi_{\mathcal{C}}$. Indeed, upon receiving $(\text{OPEN}, sid, \mathcal{P}_a)$, where $\mathcal{P}_a \in \mathcal{C}$, from $\mathcal{F}_{\mathsf{SC}}$, $\mathcal{S}_{SPA}$ simulates $\mathcal{C}_j$ using the share decryption procedure from π_{PVSS} on $\hat{\sigma}_{aj}$, obtaining $\tilde{\sigma}_{aj}, DLEQ_{aj}$. and sending $(\text{SHARE-DECRYPTION}, sid, (\hat{\sigma}_{a1}, \ldots, \hat{\sigma}_{am}), LDEI_a, CC_a, \tilde{\sigma}_{aj}, DLEQ_{aj}))$ to $\mathcal{F}_{\mathsf{SC}}$.

Fig. 19: Simulator $\mathcal{S}_{SPA}$ (Stage 4 and Recovery).

that, since $x_{w\mathbf{r}} \xleftarrow{\$} \mathbb{Z}_q$ is sampled as in Π_{SPA}, this step is indistinguishable from that as in real world execution of Π_{SPA}. On the other hand, upon receiving $(\text{LEAK-TO-WINNER}, sid, \mathbf{r}_w)$ and subsequently $(\text{ANNOUNCE?}, sid, \mathcal{P}_w)$ from $\mathcal{F}_{SPA}$, $\mathcal{S}_{SPA}$ learns the identity of the winner $\mathcal{P}_w$ and that is corrupted. Then, in case $\mathcal{P}_w$ does not send $(\text{WINNER}, sid, \mathcal{P}_w, x_{w\mathbf{r}})$ to all the other parties, $\mathcal{S}_{SPA}$ sends $(\text{NOT-ANNOUNCE}, sid,)$ to $\mathcal{F}_{SPA}$ and $\mathcal{P}_w$ will be identified as a cheater when the recovery procedure is simulated. Analogous consideration to the Theorem 1 case (Stages 2 and 3) and this approach make the view of the adversary $\mathcal{A}$ in the simulation by $\mathcal{S}_{SPA}$ indistinguishable from the real world execution.

- **3b - After First Unique Veto:** At this point, in case $\mathcal{P}_w \in \mathcal{C}$, she will not participate in the next steps of Stage 3b. Similarly, in case $\mathcal{P}_w \notin \mathcal{C}$, $\mathcal{S}_{SPA}$ simulates $\mathcal{P}_w \in \mathcal{H}$ not participating in the next steps of Stage 3b. Then, the second price b_{w_2} has to be determined. Analogous consideration to the Theorem 1 case (Stage 3) and this approach make the view of the adversary $\mathcal{A}$ in the simulation by $\mathcal{S}_{SPA}$ indistinguishable from the real world execution.
- **Stage 4 - Output:** In case one of the honest parties in the real world is the winner, $\mathcal{S}_{SPA}$ simulates $\mathcal{P}_w \in \mathcal{H}$ winning by creating a confidential trans-

action tx_{pay} and sending $(\text{OUTPUT}, sid, \mathcal{P}_w, \mathsf{tx}_{pay}, \{sig_{sk_k}(\mathcal{P}_w|b_{w_2})\}_{k\in[n]})$ to $\mathcal{F}_{\mathsf{SC}}$. However, by Lemma 1, due to the hiding property of the commitments, the distributions of $\text{com}(b_w, r_{b_w})$ and $\text{com}(change'_w, r_{change'_w})$ are indistinguishable from those in a real world execution of Π_{SPA}. Moreover, by Lemma 3, due to the zero knowledge property of range NIZKs, π_{pay} is indistinguishable from the range proof of $\mathcal{P}_w$ in the real world.
On the other hand, in the case $\mathcal{P}_w \in \mathcal{C}$, by Lemma 1, due to the binding property of the commitments, the committed values b_w and $change'_w$ cannot be changed later by $\mathcal{A}$. Moreover, by Lemma 3, due to the soundness property of the NIZKs, it is computationally hard for the adversary $\mathcal{A}$ controlling each $\mathcal{P}_a \in \mathcal{C}$ to compute the range proof π_{pay} while the bid is not in the expected range.

– **Recovery:** $\mathcal{S}_{SPA}$ simulates aborts and corresponding recovery stages if $\mathcal{A}$ deviates from the protocol following the instructions of an honest party executing Π_{SPA}. Moreover, by Lemma 4, due to the zero knowledge property of NIZKs, NW_h for each $\mathcal{P}_h \in \mathcal{H}$ are indistinguishable from the corresponding NIZKs in the real world. Finally, by Proposition 1, it is guaranteed that the shares reconstruction is valid.

Hence, the view of $\mathcal{A}$ in the real world and of $\mathcal{S}_{SPA}$'s internal copy of $\mathcal{A}$ is indistinguishable, which concludes our proof.

F Rational strategies

In this section we consider the incentives of parties in our protocol. Note that, the protocol leaks some information in first price and second price auction, which might be of help to the adversary. However, by the design of the protocol, the set of bidders is fixed after the "bid registration" phase. That is once the bid registration phase is over, even if it is required to re-run the protocol, no new bid can be submitted. Hence, although the as the time passes the participant compile more information about the winning bid, they cannot modify their bidding strategy and rejoin the system to gain from the leaked information.

In case there is a cheating party, the proposed protocols distribute the *work* and deposit of the cheating party equally among the honest parties. The literature on fair distribution is quite rich in this domain, and highly relies on the deposits (claims) that each party has [60]. However, in our setup the amount of each party deposit is secret and such methods cannot be used to re-distribute the cheating party's deposit. Also, burning the deposit is illegal. Therefore, distributing the deposit equally seems a fair solution.

We consider the utility of each party from participating in the protocol. The the utility function of a generic party $\mathcal{P}_\mathbf{i}$ in the first price auction can be defined as follows:

$$u_\mathbf{i}^{FPA}(b_1, \ldots, b_n) = \begin{cases} v_i - b_i, & \text{if } b_i > \max_{j\neq i} b_j \\ 0, & \text{if } b_i < \max_{j\neq i} b_j \end{cases}$$

and in case of the second price auction is defined as follows:

$$u_{\mathbf{i}}^{SPA}(b_1, \ldots, b_n) = \begin{cases} v_{\mathbf{i}} - \max_{j \neq i} b_j, & \text{if } b_i > \max_{j \neq i} b_j \\ 0, & \text{if } b_i < \max_{j \neq i} b_j \end{cases}$$

where $v_{\mathbf{i}}$ represents the $\mathcal{P}_{\mathbf{i}}$'s private valuation of what is at stake in the auction. It is known that in the first price auctions, the optimal bid for each rational party, depends on their beliefs regarding other party's valuations, and in the second price auction the optimal bidding for each party is to bid equal to his valuation regardless of the bidding strategy of other parties, *i.e.*, $b_i = v_i$ (see [46, 52] for the optimal bidding strategy of parties).

Note that, in case a party $\mathcal{P}_{\mathbf{i}}$ is honest, she gets her $work$ back, and if she is the winner she gets what is at stake in the auction and pays her bid $b_{\mathbf{i}}$, or if she is not the winner then she gets her entire deposit $b_{\mathbf{i}} + work$ back. Therefore, by following the protocol each rational party can guarantee a non-negative utility, *i.e.*, $u_{\mathbf{i}}(b_1, \ldots, b_n) \geq 0$. However if a party cheats then as described in the protocol the party's $work$ and deposit is distributed among honest parties. Therefore, the utility of a cheating party, regardless of whether her bid is the highest or not, is

$$u_{\mathbf{i}}(b_1, \ldots, b_n) = -(b_i + work) < 0$$

which is strictly negative. Therefore, misbehaving is a dominated strategy for each party as regardless of what other players do it always results in lower utility.