Publicly Verifiable Auctions with Privacy

Paul Germouty[1], Enrique Larraia[1], and Wei Zhang[1]

nChain
{p.germouty,e.larraia,w.zhang}@nchain.com

Abstract. Online auctions have a steadily growing market size, creating billions of US dollars in sales value every year. To ensure fairness and auditability while preserving the bidder's privacy is the main challenge of an auction scheme. At the same time, utility driven blockchain technology is picking up the pace, offering transparency and data integrity to many applications. In this paper, we present a blockchain-based first price sealed-bid auction scheme. Our scheme offers privacy and public verifiability. It can be built on any public blockchain, which is leveraged to provide transparency, data integrity, and hence auditability. The inability to double spend on a blockchain is used to prevent bid replay attacks. Moreover, our scheme can achieve non-repudiation for both bidders and the auctioneer without revealing the bids and we encapsulate this concept inside the public verification of the auction. We propose to use ElGamal encryption and Bulletproofs to construct an efficient instantiation of our scheme. We also propose to use recursive zkSNARKs to reduce the number of comparison proofs from $N-1$ to 1, where N is the number of bidders.

Keywords: Auction · Blockchain · Privacy · Public Verifiability.

1 Introduction

In the last few decades, online auctions are becoming an important vehicle to advertise and trade assets. They connect buyers and sellers dispersed geographically and allows them to perform the exchange without being in the same location physically. In traditional online auctions platforms (such as eBay), the auctioneer is entrusted to conduct the entire auction correctly. This design demands a high degree of trustworthiness on auctioneer's side. Fortunately, we can leverage a blockchain's built-in consensus and data-integrity to add transparency to the process. This can be done in two ways. The blockchain can be used as a bulletin board where data is posted publicly by bidders and the auctioneer. Afterwards, an independent party can verify the auction using the (immutable) information on the blockchain. The other possibility is to delegate the verification to the blockchain itself, yielding a self-enforcing transparent auction. In both cases, blockchains can be effectively used to verify the correctness of the auction result.

The challenge is to design a blockchain-based auction that does not compromise privacy. In a sealed-bid auction, only the winning bid is revealed by the auctioneer, and the other bids are kept private (as opposed to open-outcry auctions, where all the bids are known to everyone). The most common types of sealed-bid auctions are first and second price winning auctions.

- First-price sealed-bid auction (FPSBA). The bidder with the highest bid wins and pays the value of the bid to the seller.
- Second-price sealed-bid auction (Vickrey auctions [29]). The bidder with the highest bid wins, but pays an amount equal to the value of the second highest bid.

In this work we focus on FPSBA for simplicity, but we emphasize that with slight modifications our system can be applied to Vickrey auctions as well. We refer to [28] for more information about the different types of auctions.

1.1 Auction model

Security: Building on [19] we define security of a blockchain-based FPSBA with the following properties:

- **Bid privacy**. No information about non-winning bids can be inferred from the result (beyond that the winning bid is the highest bid).
- **Bid independence**. Bids are independent from each other. A (possibly malicious) bidder cannot construct their bid based on a bid from an honest bidder.
- **Bid binding**. After the bid phase is closed, bidders cannot change their mind and bid differently.
- **Public verifiability**. The correctness of the result (namely, the winner is the one claimed by the auctioneer) and the correct behaviour of the auctioneer, can be verified by anyone using public information only.

Public verifiability is sometimes referred as *universal verifiability*, and bid binding as *non-repudiation* or *non-cancellation* [16]. As noted in [15], these security definitions borrow from the e-voting literature, wherein individual/universal verifiability, eligibility and accountability are well-established notions.

Adversary model. To achieve bid privacy we assume the auctioneer is *semi-honest*. Specifically, they do not share private information. For the remaining security properties, the auctioneer and the bidders can be *malicious* (they may deviate from the protocol instructions and share information). We observe that an auction with bid independence implies that there cannot be replay attacks.

Trustless verification. Public verification should be conducted based only on public information. This means that neither the auctioneer nor the bidders need to disclose private information. In most cases, all public information can be fetched from the blockchain. Public verifiability,as defined above, also captures the case of verifying that a bid has been discarded by the auctioneer because it is indeed malformed (e.g. out of range). If the chosen blockchain supports contract automation and enforcement, the verification can be embedded in blockchain transactions and conducted by the blockchain network.

Complexity: We identify metrics relating to the complexity of the auction scheme.

Communication overhead (CO). The amount of data published on the blockchain by the bidders and the auctioneer. For scalability, it should be linear in the number of bidders $\mathcal{O}(\lambda N)$. Ideally, any party publishes $\mathcal{O}(\lambda)$ bits to the blockchain, where λ is the security parameter.

Rounds of interaction (blockchain latency). This can be defined as the minimum number of blocks elapsed while executing an auction. Ideally, bidders only interact with the blockchain in the bid phase, and do not participate in the result phase nor in the public verification.

Verification overhead (VO). The complexity of the public verification algorithm. This affects the feasibility and cost of self-enforced verification in the blockchain. Public verification uses on-chain information, thus a lower bound is the communication overhead.

Financial fairness: Some schemes require that dishonest behaviour should be financially penalised. In this work we do not detail how to achieve financial fairness. However, since we obtain public verifiability we can ask bidders to deposit funds in the blockchain and only return the funds if they behave as they should. There are a number of ways this can be implemented in practice. For example, using smart contracts in an Ethereum-like blockchain as in [19, 22] or using multi-signatures escrow payments in a Bitcoin-like blockchain. However, care must be taken as publicly disclosing the deposit (which is greater but not necessarily equal than the bid) may influence the strategic interaction and equilibrium bidding behaviour [27]. Even worst, it might conflict with bid privacy or bid independence.

1.2 Related work

Since the early works of Nurmi and Salomaa [26] and Franklin and Reiter [18] many auction schemes secured by cryptography have followed [4, 7, 9, 12, 19, 24, 25]. Some works rely on Yao's millionaire's problem [12], others are based on garbled circuits [25], or homomorphic encryption [1, 4, 24]. These protocols require a third party (the auctioneer) that should remain honest and does not collude with bidders. Other schemes aim to eliminate the auctioneer [9] and achieve fairness with low complexity [3, 14]. For example, in [9] Brandt exploits the homomorphic property of ElGamal encryption with a combination of discrete-log based zero-knowledge proofs (ZKP) to construct a multiparty protocol for fully-private auctions (whereby the winning bid is not public). However, all these works require some degree of interaction of bidders in the result phase. A recent survey of auction schemes [2] offers detailed descriptions of the security properties and technologies used in those works.

Blass and Kerschbaum present Strain in [7], which is an auction protocol for blockchains secure against a malicious bidder. The scheme uses Goldwasser-Micali's homomorphic encryption [20] scheme which has large ciphertexts and ZKPs of order N . While the scheme preserves bid privacy, the comparison of two bids requires interaction between the bidders. Lafourcade et. al. in [22] propose Auctionity, a scheme for open-outcry auctions (where all bids are public) using Ethereum to secure deposits, and analyse its security in a symbolic model.

In a different fashion, Galal and Youssef in [19] describe a blockchain-based FPSBA deployed in Ethereum. Bidders commit to their bids using Pedersen commitments and send the commitments to the auction contract. Bids are integers in $\mathbb{Z}_p$ where p is the size of

the Pedersen group $\mathbb{G}_p$. Only after the auction contract has been updated with all commitments, the bidders encrypt their bids and the Pedersen openings with an asymmetric public-key encryption scheme using the public key of the auctioneer and send the ciphertexts to the auctioneer. The auctioneer (which is semi-honest only for bid privacy) uses ZKPs to prove the statement "$x_w > x_i$" where x_w is the winning (highest) bid, and x_i is any other bid. To create the comparison ZK proofs they use the following implication, which holds for any positive integer $B \leq p/2$:

$$x_w, x_i, (x_w - x_i) \mod p, \in [0, B) \Rightarrow x_w > x_i \tag{1}$$

Since x_w is publicly announced, the comparison proof can be reduced to two range proofs for the statements "$x_i \in [0, B)$" and "$\Delta_i \mod p \in [0, B)$", where $\Delta_i := x_w - x_i$. For the range proof they use a non-interactive Σ-protocol by Brickwell et. al. [10]. However, the soundness of one such ZKP proof is for 1 bit, so they need to repeat the protocol λ times (per non-winning bid x_i). This introduces significant complexity and high gas fees on the chosen blockchain. As a result, they report an implementation with the security parameter heavily restricted to $\lambda = 10$. Also, their scheme uses a dispute-resolution technique that forces bidders to disclose their bids on-chain to prove their honesty when a malicious auctioneer falsely claim that their bids are invalid. Last, in [23] Bulletproofs for bid comparison is used, however all bidders need to participate in the verification phase.

1.3 Our contributions and techniques

We describe how to conduct FPSBA over *any* public blockchain. We focus on making an auction publicly verifiable, using the blockchain for an audit trail without breaching the privacy of the non-winning and honest bidders. In our scheme, the auctioneer generates *short* ZKPs (specifically, we use SNARKs) which greatly improves the complexity of the solution compared to previous work.

Summary of our main contributions We develop a generic methodology to ensure bid privacy and bid independence using any encryption scheme and non-malleable commitments. We describe a concrete and practical instantiation using ElGamal encryption [17], Bulletproofs [11] to generate the comparison proofs, and salted hashes to commit ciphertexts. In this instantiation, the auctioneer needs to generate one comparison proof per non-winning bidder, and hence the communication overhead is linear in the number of bidders. We explain how to achieve *constant* communication overhead and sub-linear verification using recursive SNARKs for the comparison proofs. Last, both instantiations have only four rounds of interactions with the chosen blockchain: two rounds for the bid phase and one round for each of the result phase and verification phase.

It is also worth noting that in our scheme the auctioneer can prove a bid is out of range without interacting with the disputed bidder. This eliminates the need of dispute-resolution mechanisms. Further, we identify a flaw in the comparison proof used by Galal and Youssef in [19]. This flaw is described in appendix A.

Our techniques We implement the bid phase in two rounds. In the first round, bidders encrypt their bids and commit to their ciphertexts with a non-malleable commitment. In the second round, the bidders open the commitments to the ciphertexts by publishing the opening and the ciphertext. The bids are kept private 'inside' the ciphertexts, and the

ciphertexts (hence the bids) are independently generated. We call this two-round protocol *sealed encryption*. As in [19], we reduce the i-th comparison proof "$x_w > x_i$" to two range proofs. Bids are encrypted with ElGamal over a cyclic group $\mathbb{G}_p$ of order p. One can derive Pedersen commitments for x_i and $\hat{\Delta}_i := (x_w - x_i) \mod p$ using the ElGamal ciphertexts ct_i and ct_w. Our main observation is that the secret key (of the auctioneer) can be seen as the *opening* information. Thus, the auctioneer can use these two Pedersen commitments as the public inputs, and his secret key as the private input to prove with Bulletproof range proof system that $\forall x_i, \hat{\Delta}_i \in [0, B)$, for upper bound $B := 2^n$.

In Section 4, we describe a generic method using recursive SNARKs to generate a single proof attesting to the validity of all statements "$x_w > x_i$", for $i \leq N, i \neq w$. The auctioneer organizes the $N-1$ ciphertexts of the non-winning bids into a Merkle tree, and recurse proof generation over the tree. We explain how to do sequential recursion and somewhat-parallel recursion during the proof generation.

Comparison with other blockchain-based auctions We compare the reviewed auction protocols with ours in Table 1. We chose to use only one security parameter λ for both security of the encryption and soundness of the argument of knowledge. N is the number of bidders, and bids are in the range $[0, 2^n - 1]$ for some integer n. In our schemes we have $3 + 1$ rounds, where the last round is for public-verifiability. The communication overhead during the bid and result phases directly affect the cost of implementing the auction on a blockchain, as most blockchains charge fees based on the size of transactions or data. In the case of an automated verification, the verification runtime is also important as it affects the cost of executing an automated contract on-chain.

Scheme	Rounds	Communication		Verification runtime	Public verifiability	
		Bid phase	Result phase	($\#\mathbb{G}_p$ operations)	Bid correctness	Result correctness
[7]	4	$\mathcal{O}(n\lambda^2)$	$\mathcal{O}(N^2\lambda)$	$\mathcal{O}(n\lambda)$	✗	✓
[19]	$2(\lambda+1)$	$\mathcal{O}(\lambda)$	$\mathcal{O}(N\lambda^2)$	$\mathcal{O}(N\lambda^2)$	✗	✓
Ours (Section 3)	3+1	$\mathcal{O}(\lambda)$	$\mathcal{O}(\lambda N \log(n))$	$\mathcal{O}(\lambda N \frac{n}{\log(n)})$	✓	✓
Ours (Section 4)	3+1	$\mathcal{O}(\lambda)$	$\mathcal{O}(\lambda \log(n))$	$\mathcal{O}(\lambda \log(n))$	✓	✓

Table 1: Asymptotic comparison of our work

2 Preliminaries

In this section we describe the building blocks that we use in our auction scheme, namely: commitments, public-key encryption and zero-knowledge proofs.

Notation. We denote a finite cyclic group of prime order p with $\mathbb{G}_p$, and vectors of group elements in bold $\mathbf{v}$. We write $x \xleftarrow{\$} D$ to mean x is picked uniformly at random from domain D. We will use λ for the security parameter, N for the number of bidders that participate in an auction, and $n \ll \log(p)$ for an upper bound of the bids $x \in [0, 2^n)$.

2.1 Public key encryption

A public-key encryption scheme PKE has four probabilistic polynomial time (PPT) algorithms:

- $\mathsf{Setup}(1^\lambda)$ takes as input the security parameter λ, and outputs the public parameters param of the scheme.
- $\mathsf{KeyGen}(\mathsf{param})$ outputs a pair of keys, a (public) encryption key pk and a (private) decryption key sk;
- $\mathsf{Encrypt}(\mathsf{pk}, m; r)$ takes as input the public key pk a plaintext m, and randomness r. It outputs a ciphertext ct.
- $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct})$ takes as input the secret key sk and a ciphertext ct. It outputs a plaintext m or a decryption error $\perp$.

ElGamal Encryption. It is a public-key encryption scheme [17] instantiated over a cyclic group $\mathbb{G}_p$ with generator g. A ciphertext consist of a pair of group elements $\mathsf{ct} = (d, e)$, and we set the plaintext space to $[0, 2^n) \subset \mathbb{Z}_p$. The decryption requires a precomputed lookup table storing pairs of the form $(x, g^x) \in [0, 2^n) \times \mathbb{G}_p$ where x is from the plaintext space. See Figure 1 for a description.

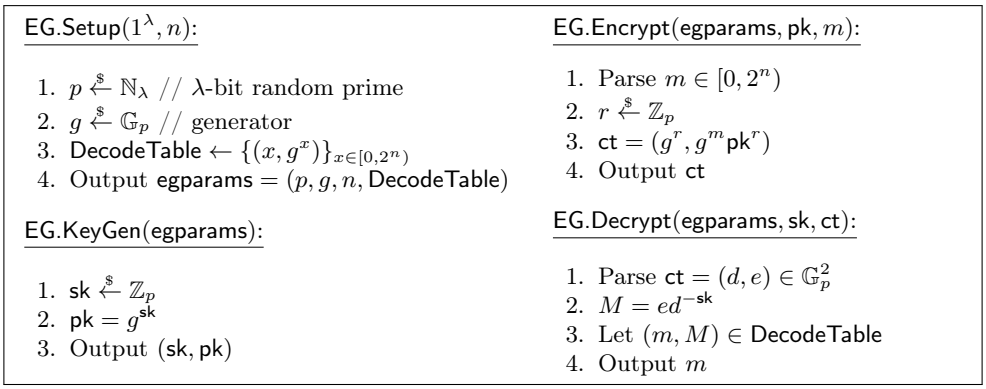

Fig. 1: ElGamal Encryption scheme

2.2 Commitments

A commitment scheme Com enables the generation of a commitment to a message, which can be used to verify the message when it is revealed with an opening. It consists of four PPT algorithms:

- $\mathsf{Setup}(1^\lambda)$ outputs the public parameters param of the scheme.
- $\mathsf{KeyGen}(\mathsf{param})$ generates a public commitment key ck.
- $\mathsf{Commit}(\mathsf{ck}, m; r)$ takes as input ck and a message m. It outputs the commitment c and the randomness r used to commit (the opening value).

– $\mathsf{VerCom}(\mathsf{ck}, \mathsf{c}, m, r)$ outputs true if message m was committed in c using r. Otherwise, it outputs false.

Intuitively, a commitment scheme is *binding* if it is infeasible to find messages $m' \neq m$ and openings r, r' such that $\mathsf{Commit}(\mathsf{ck}, m; r) = \mathsf{Commit}(\mathsf{ck}, m'; r')$. It is *hiding* if it is infeasible to infer any information about m from c without the knowledge of the opening r (the probabilities of both events are negligible in the security parameter λ).

In our auction scheme we will commit to values using a cryptographic hash function Hash as described in Figure 2. This commitment scheme does not have a commitment key, and it is well-known to be binding and hiding in the random oracle model. The range of the hash should be set to 2λ for the binding property (preimage resistance), and use a λ-bit random salt for the hiding property. This commitment scheme is also non-malleable.

$\underline{\mathsf{Setup}(1^\lambda)}$: Output $\mathsf{Hash} : \{0,1\}^* \to \{0,1\}^{2\lambda}$

$\underline{\mathsf{Commit}(m)}$:

1. $r \xleftarrow{\$} \{0,1\}^\lambda$
2. $\mathsf{c} = \mathsf{Hash}(m||r)$
3. Output (c, r)

$\underline{\mathsf{VerCom}(m, \mathsf{c}, r)}$:

1. If $\mathsf{c} = \mathsf{Hash}(m||r)$ output true
2. Else output false

Fig. 2: Salted hash commitment.

2.3 Zero-knowledge proofs

Given a fixed finite field $\mathbb{F}_p$ and an $\mathbb{F}_p$-arithmetic circuit $C : \mathbb{F}_p^n \times \mathbb{F}_p^h \to \{0,1\}$, a pre-processing, zero-knowledge, succinct, non-interactive, argument of knowledge (zkSNARK —see e.g. [6]) for the NP relation

$$\mathcal{R}_\mathcal{C} := \{(x; w) \in \mathbb{F}_p^n \times \mathbb{F}_p^h \mid C(x, w) = 1\},$$

is a triplet of algorithms $\mathsf{SNARK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ such that:

– $\mathsf{Setup}(1^\lambda, \mathcal{R}_\mathcal{C})$ takes as input a security parameter λ and the description of a circuit $\mathcal{C}$ it outputs a pair of keys pk, vk.
– $\mathsf{Prove}(\mathsf{pk}, x, w)$ takes the proving key pk, the public instance x and the private witness w as input and outputs a proof π.
– $\mathsf{Verify}(\mathsf{vk}, x, \pi)$ takes the verification key vk, the public instance x, and the proof π as input and outputs either accept or reject.

The zkSNARK is *complete* if Verify always accepts proofs π generated by Prove on inputs $(x; y) \in \mathcal{R}_C$. It is *succinct* if $|\pi| = \mathcal{O}_\lambda(1)$, and it has a succinct verifier (sometimes also referred as fully succinct) if Setup runs in time $\mathcal{O}_\lambda(|x|)$. It is zero-knowledge if no information about the witness w is leaked from the proof.

Knowledge soundness. It must be possible to efficiently extract a witness from any (possibly cheating) prover Prove^* that outputs an accepting pair (x, π). More formally, for every polynomial-time adversary Prove^*, there exists a polynomial-time extractor $\mathsf{Extract}_{\mathsf{Prove}^*}$, such that for every large-enough security parameter λ,

$$P\left[\begin{array}{l}\mathsf{Verify}(\mathsf{vk}, x, \pi) = 1 \\ (x; y) \notin \mathcal{R}_C\end{array}\middle|\begin{array}{l}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Prove}(1^\lambda, C) \\ (x, \pi) \leftarrow \mathsf{Prove}^*(\mathsf{pk}, \mathsf{vk}) \\ y \leftarrow \mathsf{Extract}_{\mathsf{Prove}^*}(\mathsf{pk}, \mathsf{vk})\end{array}\right] \leq \mathsf{negl}(\lambda).$$

Bulletproofs Bulletproofs [11] is a SNARK used to prove interval membership $x \in [0, 2^n)$. The public input is a Pedersen commitment $C = \mathsf{Commit}(\mathsf{ck}, x; r) := g^x h^r$ of the integer x under commitment key $\mathsf{ck} := (g, h) \in \mathbb{G}_p^2$. A prover convinces the verifier that a given $x \in \mathbb{Z}_p$ lies in range $[0, 2^n)$ by showing it knows the bit decomposition $\mathbf{b}$ of x. Specifically, the NP relation is:

$$\mathcal{R}_{\mathsf{BP}} = \{(g, h, C \in \mathbb{G}_p, n \in \mathbb{N}); (x, r \in \mathbb{Z}_p) \mid C = g^x h^r, x \in [0, 2^n)\} \tag{2}$$

The high-level idea is to prove in zero-knowledge that the following constraints are satisfied:

$$\langle \mathbf{b}, \mathbf{2}^n \rangle = x; \text{ and } \mathbf{b} \circ \mathbf{b}' = \mathbf{0}; \text{ and } \mathbf{b}' = \mathbf{b} - \mathbf{1}$$

Above, $\mathbf{2}^n$ is a vector of all the powers of 2 up to 2^{n-1}, and $\circ$ denotes component-wise product. The second and third constraints proves that elements in $\mathbf{b}$ are indeed bits, and the first constraint shows that $\mathbf{b}$ is the bit decomposition of x. To achieve logarithmic communication in n, satisfiability of the above constraints is reduced to a inner product argument (sound but not zero-knowledge) in which the prover commits to messages $\mathbf{m} \in \mathbb{Z}_p^{2n}$ of $2n$ elements using a (binding-only) length-reducing Pedersen vector commitment with key $\mathsf{ck} := \mathbf{g} \in \mathbb{G}_p^{2n}$, setting $\mathsf{Commit}(\mathsf{ck}, \mathbf{m}) := \prod_{i=1}^{2n} g_i^{m_i}$. The triplet of algorithms $\mathsf{BP} = (\mathsf{BP.Setup}, \mathsf{BP.Prove}, \mathsf{BP.Verify})$ is as follows:

- $\mathsf{BP.Setup}(1^\lambda, n)$ takes as input a security parameter λ and the description of the range $[0, 2^n)$. It outputs a pair of keys $\mathsf{pk_{bp}} = \mathsf{vk_{bp}} = \mathbf{g} \in \mathbb{G}_{p(\lambda)}^{2n}$.
- $\mathsf{BP.Prove}(\mathsf{pk_{bp}}, (g, h, C), (x, r))$ takes the proving key $\mathsf{pk_{bp}}$, the public instance (g, h, C) and the private witness (x, r) and outputs a proof π_{bp}.
- $\mathsf{BP.Verify}(\mathsf{vk_{bp}}, (g, h, C), \pi_{\mathsf{bp}})$ takes the verification key $\mathsf{vk_{bp}}$, the public instance (g, h, C), and the proof π_{bp}, and it outputs accepts or rejects.

The soundness of Bulletproofs relies on the assumption that there is no known relationship between the group elements $\mathbf{g}, g, h$. Recall $\mathbf{g}$ is the Vector Pedersen key used to commit to internal messages in the proving and verification algorithms of BP, and g, h is the Pedersen key (part of the public instance) used to commit to x in relation $\mathcal{R}_{\mathsf{BP}}$ — see Equation (2). The publicly-verifiable correctness of the result of our auction scheme will rely on the following theorem proved in [11].

Theorem 1 (Corollary 2 of [11], informal). *If there is no known relationship between the group elements $g, h, \mathbf{g}$, then BP with $\mathsf{pk_{bp}} = \mathsf{vk_{bp}} = \mathbf{g}$ is knowledge sound.*

Recursive proof composition. This is useful to prove that a function is applied iteratively $x_1 = f(x_0), \ldots, x_n = f(x_{n-1})$. Proving the correct iteration of f can be done with proof composition. A proof π_n for the statement "$x_n = f(x_{n-1})$" also attests to the existence of valid proofs π_i for the statements "$x_i = f(x_{i-1})$" for each $i = 1 \ldots, n-1$. In a recursive SNARK_r, the verification logic of the base SNARK_b is implemented as part of the prover. Thus, if the base circuit is "$\mathcal{C}_b(x_i, x_{i-1}) = 1$ iff $x_i = f(x_{i-1})$", the recursive algorithm Prove_r proves satisfiability of the augmented circuit:

$$\text{“}\mathcal{C}_r((x_i, \mathsf{vk}_b), (x_{i-1}, \pi_{i-1})) = 1 \text{ iff } x_i = f(x_{i-1}) \wedge \mathsf{Verify}_b(\mathsf{vk}_b, x_{i-1}, \pi_{i-1}) = \mathsf{true}\text{”}.$$

The main benefit of recursive proofs is that the prover can be parallelised. Of course, it also reduces the number of proofs that need to be transmitted from $N-1$ (using a non-recursive SNARK) to just one. Proof composition was first constructed using cycles of pairing-friendly curves [5] and can be recursed assuming the base verifier is succinct (it runs in logarithmic time). Further works [8, 21], reduce the recursion overhead without requiring a succinct base verifier.

3 Description of our scheme

In this section we describe the details of our auction scheme. The interactions after setup, between the blockchain and the participants of the auction are described in Figure 3 and Figure 4. The verification of the auction can be conducted by an independent party or automated on the blockchain. Our auction scheme has four phases: setup, bid, result, and

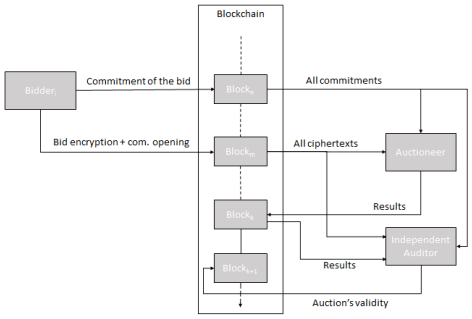

Fig. 3: Auction schemes interaction with independent public verification

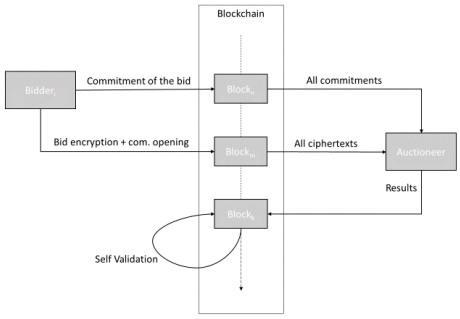

Fig. 4: Auction schemes interaction with automated verification

public verification. It can be implemented with appropriate choices of commitments, public-key encryption and ZKPs. We give a concrete instantiation by setting the commitment to salted hashes, encryption to ElGamal [17] and the ZKP to Bulletproof [11]. The details of these building blocks can be found in Section 2. In Figure 8 we describe the implementation of our auction scheme.

3.1 Setup Phase

In this phase, the auctioneer sets time bounds T_b, T_r, T_v for each subsequent phase (e.g. given by block heights). It then generates the following scheme parameters:

- Select a cyclic group $\mathbb{G}_p$ of prime order $p = p(\lambda)$.
- Set the upper bound of the bids to $n < \log(p) - 1$. Thus the bid range is $[0, 2^n)$ for $2^n < p/2$.
- Generate $2n+1$ group elements $g, \mathbf{g} \in \mathbb{G}_p^{2n+1}$. These group elements are generated in a publicly-verified way to ensure no discrete-log relation is known. The proving key (and verification key) of Bulletproofs is set to $\mathbf{g}$. The group generator of ElGamal is set to g.
- Compute the ElGamal decoding table $\mathsf{DecodeTable} = \{(x, g^x)\}_{x \in [0,2^n)}$ and his secret/public key pair $(\mathsf{pk}_A, \mathsf{sk}_A)$.

Then, the auctioneer publishes $(n, p, g, \mathbf{g}, \mathsf{pk}_A, T_b, T_r, T_v)$ and the description of the goods to the blockchain. It keeps the decryption key sk_A private.

3.2 Bidding Phase

We design a new protocol for the bid phase called *sealed encryption* and described in Figure 5. This new protocol allows for a message to be committed and revealed to an intended recipient at a later stage. It uses two cryptographic primitives: encryption and commitment as defined in Section 2. For the auction use case, the message is the bid and the intended recipient is the auctioneer. We make use of the security of Bitcoin-like blockchains to prevent man-in-the-middle attacks. Our construction can be generalized to any blockchain of which transactions are secure against man-in-the-middle attacks. More concretely, in the bid phase, we instantiate a sealed encryption using ElGamal encryption and salted hash:

Commit phase: the bidder encrypts his bid x with ElGamal using the public key pk_A of the auctioneer. The ciphertext ct is concatenated with a transaction identifier txid_0, corresponding to a transaction tx_0 spendable by the bidder. Then $\mathsf{ct}||\mathsf{txid}_0$ is committed in c using randomness r. Finally, c is pushed on the blockchain by embedding it in a transaction tx_1 that spends tx_0.

Reveal phase: the bidder publishes his ciphertext ct and the opening randomness r on the blockchain by embedding them in a transaction tx_2 that spends tx_1.

3.3 Result phase

In this phase, the auctioneer identifies the winning bid x_w and announces it publicly in the blockchain. The auctioneer does the following:

- Obtain and decrypt all ElGamal ciphertexts $\{\mathsf{ct}_i\}_{i \leq N}$ from the bid phase using his secret key sk_A.
- Discard incorrect bids: If an opening to a ciphertext ct_i is incorrect add i to a list of incorrect openings $\mathsf{L}_{\mathsf{BadComm}}$. If it decrypts to group element X_i that cannot be decoded (bid out of range) add i to a list of bad encryptions $\mathsf{L}_{\mathsf{BadEnc}}$. The auctioneer disqualify all bidders in $\mathsf{L}_{\mathsf{BadComm}} \cup \mathsf{L}_{\mathsf{BadEnc}}$. Else, he adds i to a list of honest bidders L_{HB}.

$\underline{\mathsf{SE.Commit}(\mathsf{param}, \mathcal{M}, \mathsf{pk})}$:

1. Pick an unspent transaction txid_0
2. Compute $\mathsf{ct} = \mathsf{Encrypt}(\mathsf{param}, \mathsf{pk}, m)$
3. Pick $r \xleftarrow{\$} \mathbb{Z}_p$
4. Compute $\mathsf{c} = \mathsf{Commit}(\mathsf{ck}, \mathsf{ct}||\mathsf{txid}_0; r)$
5. Embed c in the transaction txid_1 spending the transaction txid_0

$\underline{\mathsf{SE.Reveal}(\mathsf{ct}, r, \mathsf{txid}_1)}$:

1. Embed ct, txid_0 and r in the transaction txid_2 spending txid_1.

$\underline{\mathsf{SE.Verify}(\mathsf{ct}, r, \mathsf{c}, \mathsf{txid}_2)}$:

1. Verify c lies inside the parent transaction of txid_2 and child transaction of txid_0.
2. Output $\mathsf{VerCom}(\mathsf{ck}, \mathsf{c}, \mathsf{ct}||\mathsf{txid}_0; r)$

$\underline{\mathsf{SE.Open}(\mathsf{ct}, r, \mathsf{c}, \mathsf{txid}_2, \mathsf{sk})}$:

1. Run $\mathsf{SE.Verify}(\mathsf{param}, \mathsf{ct}, r, \mathsf{c}, \mathsf{txid}_2)$ if it ouputs 0 then output $\perp$
2. $m \leftarrow \mathsf{Decrypt}(\mathsf{param}, \mathsf{sk}, \mathsf{ct})$
3. Output m

Fig. 5: Sealed Encryption Protocol

- Identify the highest bid x_w from the set of honest bids $\mathsf{L_{HB}}$.
- Prove correctness of his computations. He creates a correct decryption proof $\pi_{\mathsf{dec},w}$ to prove ct_w decrypts to x_w, and (at most) $N-1$ comparison proofs $\pi_{\mathsf{cmp},i}$ to prove that $x_w > x_i$ for each $i \neq w, i \in \mathsf{L_{HB}}$. He also creates proofs $\pi_{\mathsf{dec},i}$ of correct decryption to out of range bid X_i for each $i \in \mathsf{L_{BadEnc}}$. (See next paragraphs for more information.)
- Publish to the blockchain the winner bid x_w, the list of ZKP proofs $\mathsf{L_{zkp}} = \{\pi_{\mathsf{dec},w}, \{\pi_{\mathsf{cmp},h}\}_{h\neq w, h\in \mathsf{L_{HB}}}, \{\pi_{\mathsf{dec},c}\}_{c\in \mathsf{L_{BadEnc}}}\}$, and the list of honest bidders $\mathsf{L_{HB}}$ and disqualified bidders $\mathsf{L_{BadComm}} \cup \mathsf{L_{BadEnc}}$.

Correct decryption proof. This is essentially a proof of equal discrete logarithms. Let the auctioneer ElGamal public key $\mathsf{pk}_A = h = g^{\mathsf{sk}_A}$, and let the ElGamal ciphertext $\mathsf{ct}_w = (d_w, e_w) = (g^r, g^{x_w} h^r)$. If the ciphertext decrypts to x_w under secret key sk_A, then it holds $e_w x_w^{-1} = h^r = d_w^{\mathsf{sk}_A}$. Thus, the auctioneer proves in zero-knowledge that the group elements $e_w x_w^{-1}$ and h have the same discrete logarithm sk_A in basis d_w, g respectively. We use the Σ-protocol of Chaum and Pedersen [13] to prove equality of discrete logs on public basis. For completeness, we describe the non-interactive version $\mathsf{CDEC} = (\mathsf{CDEC.Prove}, \mathsf{CDEC.Verify})$ in Figure 6.

$\underline{\mathsf{CDEC.Prove}((g, \mathsf{pk}_A, \mathsf{ct}, x), \mathsf{sk}_A)}$:

1. Parse $\mathsf{ct} = (d, e) \in \mathbb{G}_p^2$
2. $s \xleftarrow{\$} \mathbb{Z}_p$
3. $(a, b) = (g^s, d^s)$
4. $c = \mathsf{Hash}(g, \mathsf{pk}_A, x, \mathsf{ct}, a, b) \in \mathbb{Z}_p$
5. $z = s + c\mathsf{sk}_A$
6. Output $\pi_{\mathsf{dec}} = (c, z)$

$\underline{\mathsf{CDEC.Verify}((g, \mathsf{pk}_A, \mathsf{ct}, x), \pi_{\mathsf{dec}})}$:

1. Parse $\mathsf{ct} = (d, e) \in \mathbb{G}_p^2$
2. Parse $\pi_{\mathsf{dec}} = (c, z) \in \mathbb{Z}_p^2$
3. $a = g^z \mathsf{pk}_A^{-c}$,
4. $b = d^z (e g^{-x})^{-c}$
5. If $c = \mathsf{Hash}(g, \mathsf{pk}_A, \mathsf{ct}, x, a, b)$ output true
6. Else, output false

Fig. 6: Non-interactive sigma protocol to prove correct decryption of ct to x using decryption key sk_A. Based on equality of discrete logs [13].

Comparison proof. The auctioneer needs to prove that for each $i \neq w$ the bid x_w encrypted in the ElGamal ciphertext $\mathsf{ct}_w = (d_w, e_w)$ is greater than the bid x_i encrypted in $\mathsf{ct}_i = (d_i, e_i)$. As in [19], we reduce bid comparison to checking interval membership, namely that $x_w, x_i, \hat{\Delta}_i := (x_w - x_i) \mod p \in [0, B)$ for some fixed bound $B := 2^n \leq p/2$ —see implication (1). The NP relation $\mathcal{R}_{\mathsf{CMP}}$ for bid comparison is then as follows:

$$\mathcal{R}_{\mathsf{CMP}} = \left\{ \begin{array}{l|l} g, d_w, e_w, d_i, e_i \in \mathbb{G}_p, n \in \mathbb{N}; & e_w = g^{x_w} d_w^{\mathsf{sk}_A} \\ x_w, x_i, \mathsf{sk}_A \in \mathbb{Z}_p & e_i = g^{x_i} d_i^{\mathsf{sk}_A} \\ & \hat{\Delta}_i = x_w - x_i \mod p \\ & x_i, \hat{\Delta}_i \in [0, 2^n - 1] \end{array} \right\} \tag{3}$$

We use Bulletproofs [11] as a building block. Recall from section 2.3 that Bulletproofs take as public input a Pedersen commitment $C = g^x h^r \in \mathbb{G}_p$, and as private input the integer $x \in [0, 2^n)$ and the opening $r \in \mathbb{Z}_p$. Our observation is that an ElGamal ciphertext $\mathsf{ct} = (d, e) \in \mathbb{G}_p^2$, encrypted with public key $\mathsf{pk} = g^{\mathsf{sk}}$ can be seen as a Pedersen commitment under commitment key $\mathsf{ck} = (g, d)$. More precisely, the second ciphertext component e can be seen as a Pedersen commitment with opening the secret key sk:

$$\mathsf{ct} = (d, e) = (g^r, g^x h^r) = (g^r, g^x g^{\mathsf{sk} r}) = (d, g^x d^{\mathsf{sk}}) \tag{4}$$

The above equation means that the auctioneer can use the ciphertext $\mathsf{ct}_i = (d_i, e_i)$ and his private key sk_A to prove with Bulletproofs that what the ciphertext decrypts to (the bid x_i) is in the valid range $[0, 2^n)$. Since ElGamal is additively homomorphic, the auctioneer can also derive a ciphertext for the difference $\hat{\Delta}_i = (x_w - x_i) \mod p$ setting

$$\mathsf{ct}_{\hat{\Delta}_i} = (d_{\hat{\Delta}_i}, e_{\hat{\Delta}_i}) := (d_w d_i^{-1}, e_w e_i^{-1}) = (g^{r_w - r_i}, g^{\hat{\Delta}_i} \mathsf{pk}_A^{r_w - r_i}) = (d_{\hat{\Delta}_i}, g^{\hat{\Delta}_i} d_{\hat{\Delta}_i}^{\mathsf{sk}_A}) \tag{5}$$

and prove $\hat{\Delta}_i \in [0, 2^n)$. In Figure 7 we detail the snark $\mathsf{CMP} = (\mathsf{CMP.Setup}, \mathsf{CMP.Prove}, \mathsf{CMP.Verify})$ to prove correct bid comparison. Thus, it proves that $((g, \mathsf{ct}_w, \mathsf{ct}_i, n); (x_w, x_i, \mathsf{sk}_A)) \in \mathcal{R}_{\mathsf{CMP}}$.

Soundness of the comparison proof Due to lack of space we just give the intuitions of why the comparison proofs $\pi_{\mathsf{cmp},i}$ are sound. We just have to argue for honest bidders. If the i-th non-winning bidder is honest, he will not reveal the encryption randomness r_i used to encrypt $\mathsf{ct}_i = (d_i, e_i) = (g^{r_i}, g^{x_i} \mathsf{pk}_A^{r_i})$ to the auctioneer. Thus, the auctioneer does not known the discrete-log relationship between g, and $d_i = g^{r_i}$, where g is the ElGamal generator. Now, because no one knows a discrete-log relationship between the group elements $g, \mathbf{g}$ that form the verification key $\mathsf{vk}_{\mathsf{cmp}}$ (they are generated in a publicly verifiable way), it is not difficult to see that $g, d_i, \mathbf{g}$ are also independent elements from the point of view of the auctioneer. Using Theorem 1, the soundness of the bulleproof π_i generated in step 5 of algorithm $\mathsf{CMP.Prove}$ of Figure 7 is guaranteed.

The same applies when arguing the soundness of the bulleptroof $\pi_{\hat{\Delta}_i}$ generated in step 6 of $\mathsf{CMP.Prove}$. The elements $g, d_{\hat{\Delta}_i}, \mathbf{g}$ are independent for the auctioneer, where $d_{\hat{\Delta}_i}$ is the first component of the (homomorphically derived) ciphertext $\mathsf{ct}_{\hat{\Delta}_i}$ encrypting $\hat{\Delta}_i := (x_w - x_i) \mod p$. Indeed if r_i is unknown, then g and $d_{\hat{\Delta}_i} = g^{r_w + r_i}$ are independent elements for the point of view of the auctioneer (see Equation (5)). Therefore, the i-th comparison proof $\pi_{\mathsf{cmp},i} = (\pi_i, \pi_{\hat{\Delta}_i})$ is sound.

$\underline{\mathsf{CMP.Setup}(1^\lambda, n)}$:

1. $g, \mathbf{g} \xleftarrow{\$} \mathbb{G}_{p(\lambda)}^{2n+1}$ // Independent generators.
2. Output $\mathsf{pk}_{\mathsf{cmp}} = \mathsf{vk}_{\mathsf{cmp}} = (g, \mathbf{g})$ // $\mathbf{g}$ for Bulletproofs, and g for ElGamal.

$\underline{\mathsf{CMP.Prove}(\mathsf{pk}_{\mathsf{cmp}}, (\mathsf{ct}_w, \mathsf{ct}_i), (x_w, x_i, \mathsf{sk}_A))}$:

1. Parse $\mathsf{pk}_{\mathsf{cmp}} = (g, \mathbf{g})$
2. Parse $\mathsf{ct}_w = (d_w, e_w), ct_i = (d_i, e_i)$
3. Set $\hat{\Delta} = x_w - x_i \mod p$
4. Set $d_{\hat{\Delta}} = d_w d_i^{-1}, e_{\hat{\Delta}} = e_w e_i^{-1}$
5. $\pi_i \leftarrow \mathsf{BP.Prove}(\mathbf{g}, (g, d_i, e_i, n), (x_i, \mathsf{sk}_A))$
6. $\pi_{\hat{\Delta}} \leftarrow \mathsf{BP.Prove}(\mathbf{g}, (g, d_{\hat{\Delta}}, e_{\hat{\Delta}}, n), (\hat{\Delta}, \mathsf{sk}_A))$
7. Output $\pi_{\mathsf{cmp},i} = (\pi_i, \pi_{\hat{\Delta}})$

$\underline{\mathsf{CMP.Verify}(\mathsf{vk}_{\mathsf{cmp}}, (\mathsf{ct}_w, \mathsf{ct}_i), \pi_{\mathsf{cmp},i})}$:

1. Parse $\mathsf{vk}_{\mathsf{cmp}} = (g, \mathbf{g})$
2. Parse $\mathsf{ct}_w = (d_w, e_w), ct_i = (d_i, e_i)$
3. Parse $\pi_{\mathsf{cmp},i} = (\pi_i, \pi_\Delta)$
4. Set $d_{\hat{\Delta}} = d_w d_i^{-1}, e_{\hat{\Delta}} = e_w e_i^{-1}$
5. $b_i = \mathsf{BP.Verify}(\mathbf{g}, (g, d_i, e_i, n), \pi_i)$
6. $b_{\hat{\Delta}} = \mathsf{BP.Verify}(\mathbf{g}, (g, d_{\hat{\Delta}}, e_{\hat{\Delta}}, n), \pi_{\hat{\Delta}})$
7. If $b_i = 0, b_{\hat{\Delta}} = 0$ output 0 (accept)
8. Else output 1 (reject).

Fig. 7: SNARK to compare two bids x_w, x_i encrypted in ElGamal ciphertexts $\mathsf{ct}_w, \mathsf{ct}_i$. Internally, it uses Bulletproofs [11].

3.4 Public Verification

In this phase, the auditor verifies the correct behaviour of all bidders and the auctioneer. Namely, they obtain and verify correct openings of all commitments c_i to ciphertexts ct_i for $i \leq N$, and that the ZKP proofs of the auctioneer are valid. Note that the verifier can be automated on the blockchain in some cases.

The verifier maintains a running list L_{HB} of honest bids, initially set to all bidders. It does the following:

- *Verify correct openings of commitments to ciphertexts.* Use the algorithm VerCom which in our instantiation takes as input the commitment c_i, the randomness r_i and the ciphertext concatenated with transaction id $\mathsf{ct}_i||\mathsf{txid}_i$ and output true if $\mathsf{c}_i = \mathsf{Hash}(\mathsf{ct}_i||\mathsf{txid}_i||r_i)$. If the check fails, remove i from the list of honest bidders L_{HB}.
- *Verify incorrect encrypted bids* For each bidder $c \in \mathsf{L}_{\mathsf{HB}}$ for which the auctioneer claims ct_c decrypts to incorrect X_c, verify the correct decryption proof $\pi_{\mathsf{dec},c}$ on public input $(g, \mathsf{pk}_A, \mathsf{ct}_w, X_c)$. Here g is the ElGamal generator from setup and pk_A is the public key of the auctioneer (can be found on the blockchain). Then, verify X_c is not in the decoding table $\mathsf{DecodeTable}$. Last, remove c from the list of honest bidders L_{HB}.
- *Verify correct decryption of* ct_w *to winning bid* x_w. First check $w \in \mathsf{L}_{\mathsf{HB}}$. Then, run the verification algorithm $\mathsf{CDEC.Verify}$ on public input $(g, \mathsf{pk}_A, \mathsf{ct}_w, g^{x_w})$ and proof $\pi_{\mathsf{dec},w}$.
- *Verify* x_w *is the highest bid.* For each non-winning bidder $i \neq w, i \in \mathsf{L}_{\mathsf{HB}}$, run the verification algorithm $\mathsf{CMP.Verify}$ on public input $(\mathsf{ct}_w, \mathsf{ct}_i)$ and proof $\pi_{\mathsf{cmp},i}$ using the verification key $\mathsf{vk}_{\mathsf{cmp}}$. Here $\mathsf{vk}_{\mathsf{cmp}}$ was made publicly available by the auctioneer during setup, ct_w is the ciphertext of the winning bid (for which correct decryption is checked), and for each $i \neq w$ in the honest list, the ciphertext ct_i was published in the bid phase.

An invalid commitment opening to a ciphertext ct_i or an out of range ciphertext ct_i means the i-th bidder is dishonest, so his bid has been discarded correctly. An invalid or missing

ZKP proof means the auctioneer is dishonest. If all the ZKP proofs are valid, the auction is valid and the winner wins the auction item and pays the amount x_w.

4 Reducing the number of comparison proofs

The approach described in Section 3 requires the auctioneer to generate $N-1$ comparison proofs, one per non-winning bidder. In this section we explain a generic approach that leverages recursive SNARKs to produce a single comparison proof attesting for the $N-1$ comparisons. Note that reducing the number of comparison proofs minimizes the overall proof size, and simplifies the process of verifying the correctness of the auction result. Another interesting effect of the proof size reduction is the cost of publishing the proof on the blockchain. Since it is of size $\mathcal{O}(\log(n))$ the modified scheme is very scalable.

Our proposal allows to use any encryption scheme $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ for which it is possible to prove in zero-knowledge correct keypair generation and correct decryption (namely, without revealing the decryption key). We define the comparison predicate $\mathcal{C}_{\mathsf{CMP}}$ in Figure 9. In a nuthsell, the circuit enforces $x_w > x_i$ and that x_i can be decrypted from an input ciphertext ct_i using the secret key sk_A corresponding to the auctioneer public key pk_A.

We also need to ensure recursive comparison is done exactly on the bids encrypted in the ciphertexts $\{\mathsf{ct}_i\}_{i \neq w}$ posted in the blockchain in the bid phase. We propose two ways of enforcing this.

Sequential recursion. For simplicity, below we assume $N - 1 = 2^d$. We see the 2^d ciphertexts as the leaves of a Merkle tree, whose root is given as public input. An extra gadget is added to the comparison predicate $\mathcal{C}_{\mathsf{CMP}}$: it receives the root of the tree as public input, and the Merkle proof for the i-th leaf as private input. It enforces ct_i is in the tree using the Merkle proof. This approach has the disadvantage that proof generation is sequential. Thus, the auctioneer cannot generate $\pi_{\mathsf{cmp},i}$ at the same time than $\pi_{\mathsf{cmp},i-1}$.

Parallel recursion. Now we assume $N-1 = 2^d - 1$. To be able to batch proof generation we see the ciphertexts as the $2^d - 1$ root nodes of a Merkle tree of depth $d-1$. We define the hash at node i as $h_i = \mathsf{Hash}(\mathsf{ct}_i, h_{i,0}, h_{i,1})$, where $h_{i,b}$ denotes the hashes of the two children of node i. The extra gadget of $\hat{\mathcal{C}}_{\mathsf{CMP}}$ this time receives as public input the hash h_i, and as private inputs, the two child ciphertexts and the two child hashes. It enforces correct hash generation. The downside with respect the previous approach is that the complexity is increased because now two child proofs must be verified in the recursive circuit $\hat{\mathcal{C}}_{\mathsf{CMP}}$ (instead of one proof verification as in the sequential recursion). However, note that to generate proofs at layer k of the tree, the recursive prover only needs two proofs from the previous layer $k-1$, and all the proofs in the same layer can be parallelised. Thus, we can parallelise proof generation in batches of $2^{d-1}, 2^{d-2}, \ldots, 2$ sizes.

References

1. Abe, M., Suzuki, K.: M+1-st price auction using homomorphic encryption. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 115–124. Springer, Heidelberg, Germany, Paris, France (Feb 12–14, 2002). https://doi.org/10.1007/3-540-45664-3_8

$\mathsf{Auction.Setup}(1^\lambda)$: // Auctioneer

1. Set time bounds T_b, T_r, T_v
2. Set $n \in \mathbb{N}$ // Upper bound for bids $x \in [0, 2^n)$
3. $\mathsf{pk_{cmp}} = \mathsf{vk_{cmp}} = (g, \mathbf{g}) \leftarrow \mathsf{CMP.Setup}(1^\lambda, n)$ // Transparent setup (see Figure 7)
4. ElGamal setup: // Equivalent to $\mathsf{EG.Setup}$ (see Figure 1)
   (a) $\mathsf{DecodeTable} \leftarrow \{(x, g^x)\}_{x \in [0,2^n)} \subset \mathbb{Z}_p \times \mathbb{G}_p$
   (b) Set $\mathsf{egparams} = (p, g, n)$ and $\mathsf{DecodeTable}$
5. $(\mathsf{pk}_A, \mathsf{sk}_A) \leftarrow \mathsf{EG.KeyGen}(g)$
6. Publish $\mathsf{param} := (n, \mathsf{egparams}, \mathsf{vk_{cmp}}, \mathsf{pk}_A, T_b, T_r, T_v)$ on the blockchain

$\mathsf{Auction.Bid}(\mathsf{egparams}, \mathsf{pk}_A, x)$: // Bidders. On input bid $x \in [0, 2^n)$:

1. $(\mathsf{c}, \mathsf{ct}, r, \mathsf{tx}_1) \leftarrow \mathsf{SE.Commit}(x, \mathsf{pk}_A)$ // tx_1 hs commitment c
2. Publish tx_1 on the blockchain
3. $\mathsf{tx}_2 \leftarrow \mathsf{SE.Reveal}(\mathsf{ct}, r, \mathsf{txid}_1)$ // tx_2 spends tx_1 and contains encrypted bid ct and commitment opening r.
4. Publish tx_2 on the blockchain

$\mathsf{Auction.Result}(\mathsf{egparams}, \mathsf{pk_{cmp}}, \mathsf{L_{ct}}, \mathsf{L_{com}}, \mathsf{L_{op}}, \mathsf{sk}_A)$: // Auctioneer. Let lists of commitments, openings and ciphertexts $\mathsf{L_{com}} = \{\mathsf{c}_i\}_{1 \le i \le N}$, $\mathsf{L_{op}} = \{r_i\}_{1 \le i \le N}$, $\mathsf{L_{ct}} = \{\mathsf{ct}_i, \mathsf{tx}_{2,i}\}_{1 \le i \le N}$ from the bid phase (before time T_b)

1. $\mathsf{L_{HB}}, \mathsf{L_{BadComm}}, \mathsf{L_{BadEnc}} \leftarrow \emptyset$ // List of honest bidders, bad commitments, and bad encryptions
2. Open sealed bids from bidders. For $i \in [0, N]$:
   (a) $(b_i, x_i) \leftarrow \mathsf{SE.Open}(\mathsf{ct}_i, r_i, \mathsf{c}_i, \mathsf{tx}_{2,i}, \mathsf{sk}_A)$ // Verify opening and decrypt
   (b) If $b_i = \mathsf{true}$ add i to $\mathsf{L_{HB}}$.
   (c) Else, if c_i cannot be decommitted, add i to $\mathsf{L_{BadComm}}$. Else ct_i decrypts to out of range bid X_i, add i to $\mathsf{L_{BadEnc}}$
3. Let $x_w = \max_{i \in \mathsf{L_{HB}}}\{x_i\}$ // The winning bid
4. Create ZKP proofs:
   (a) $\pi_{\mathsf{dec}} = \mathsf{CDEC.Prove}((\mathsf{egparams}, \mathsf{pk}_A, \mathsf{ct}_w, x_w), \mathsf{sk}_A)$ // Correct decryption to winning bid x_w
   (b) $\pi_{\mathsf{cmp},i} = \mathsf{CMP.Prove}(\mathsf{pk_{cmp}}, (\mathsf{ct}_w, \mathsf{ct}_i), (x_w, x_i, \mathsf{sk}_A))$ // Correct comparison for $i \in \mathsf{L_{HB}} \backslash \{w\}$
   (c) $\pi_{\mathsf{dec},i} = \mathsf{CDEC.Prove}((\mathsf{egparams}, \mathsf{pk}_A, \mathsf{ct}_i, X_i), \mathsf{sk}_A)$ // Correct decryption to bid X_i out of range for $i \in \mathsf{L_{BadEnc}}$
   (d) Set list $\mathsf{L_{zkp}} := (\pi_{\mathsf{dec}}, \{\pi_{\mathsf{cmp},i}\}_{i \in \mathsf{L_{HB}} \backslash \{w\}}, \{X_i, \pi_{\mathsf{dec},i}\}_{i \in \mathsf{CBList}})$
5. Publish winning bid x_w and lists $\mathsf{L_{HB}}, \mathsf{L_{BadComm}}, \mathsf{L_{BadEnc}}, \mathsf{L_{zkp}}$ to the Blockchain

$\mathsf{Auction.Verify}(\mathsf{egparams}, \mathsf{vk_{cmp}}, \mathsf{L_{ct}}, \mathsf{L_{op}}, x_w, \mathsf{L_{HB}}, \mathsf{L_{BadComm}}, \mathsf{L_{BadEnc}}, \mathsf{L_{zkp}})$ // Independent party or Blockchain. On input $\mathsf{L_{com}}, \mathsf{L_{op}}, \mathsf{L_{ct}}$ from the bid phase, winning bid x_w, and lists $\mathsf{L_{HB}}, \mathsf{L_{BadComm}}, \mathsf{L_{BadEnc}}, \mathsf{L_{zkp}}$, from the result phase. If any check fails the auctioneer is dishonest, else the result is valid:

1. Check $[1, N] = \mathsf{L_{HB}} \cup \mathsf{L_{BadComm}} \cup \mathsf{L_{BadEnc}}$
2. Verify openings:
   (a) Check $\mathsf{True} = \mathsf{SE.Verify}(\mathsf{ct}_i, r, \mathsf{c}, \mathsf{tx}_{2,i})$ // for $i \in \mathsf{L_{HB}} \cup \mathsf{L_{BadEnc}}$
   (b) Check $\mathsf{False} = \mathsf{SE.Verify}(\mathsf{ct}_i, r, \mathsf{c}, \mathsf{tx}_{2,i})$ // for $i \in \mathsf{L_{BadComm}}$
3. Check $w \in \mathsf{L_{HB}}$ for winning bid x_w
4. Verify ZKPs:
   (a) Check $\mathsf{True} = \mathsf{CDEC.Verify}((\mathsf{egparams}, \mathsf{pk}_A, \mathsf{ct}_w, x_w), \pi_{\mathsf{dec}})$ // Correct decryption for x_w
   (b) Check $\mathsf{True} = \mathsf{CMP.Verify}(\mathsf{vk_{cmp}}, (x_w, \mathsf{ct}_i), \pi_{\mathsf{cmp},i})$ // for $i \in \mathsf{L_{HB}}, i \neq w$
   (c) Check $\mathsf{True} = \mathsf{CDEC.Verify}((\mathsf{egparams}, \mathsf{pk}_A, \mathsf{ct}_i, X_i), \pi_{\mathsf{dec},i})$ // ct_i decrypts to X_i for $i \in \mathsf{L_{BadEnc}}$
   (d) Check $X_i \notin \mathsf{DecodeTable}$ // X_i out of range for $i \in \mathsf{L_{BadEnc}}$

Fig. 8: Auction Scheme with ElGamal encryption and Bulletproofs

Public input Winning bid x_w, auctioneer public key pk_A, i-th ciphertext ct_i.
Private input i-th bid x_i, auctioneer secret key sk_A, randomness r for key generation.
Steps Enforce the following.

1. $x_w > x_i$
2. $x_i = \mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}_i)$
3. $(\mathsf{pk}_A, \mathsf{sk}_A) = \mathsf{KeyGen}(\mathsf{param}; r)$

Fig. 9: Comparison predicate $\mathcal{C}_{\mathsf{CMP}}$ for a generic encryption scheme.

2. Alvarez, R., Nojoumian, M.: Comprehensive survey on privacy-preserving protocols for sealed-bid auctions. Comput. Secur. **88** (2020). https://doi.org/10.1016/j.cose.2019.03.023, `https://doi.org/10.1016/j.cose.2019.03.023`
3. Bag, S., Hao, F., Shahandashti, S.F., Ray, I.G.: SEAL: sealed-bid auction without auctioneers. IEEE Trans. Inf. Forensics Secur. **15**, 2042–2052 (2020)
4. Baudron, O., Stern, J.: Non-interactive private auctions. In: Syverson, P.F. (ed.) FC 2001. LNCS, vol. 2339, pp. 364–378. Springer, Heidelberg, Germany, Grand Cayman, British West Indies (Feb 19–22, 2002)
5. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Scalable zero knowledge via cycles of elliptic curves. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 276–294. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2014). https://doi.org/10.1007/978-3-662-44381-1_16
6. Bitansky, N., Canetti, R., Chiesa, A., Goldwasser, S., Lin, H., Rubinstein, A., Tromer, E.: The hunting of the SNARK. Journal of Cryptology **30**(4), 989–1066 (Oct 2017). https://doi.org/10.1007/s00145-016-9241-9
7. Blass, E.O., Kerschbaum, F.: Strain: A secure auction for blockchains. In: López, J., Zhou, J., Soriano, M. (eds.) ESORICS 2018, Part I. LNCS, vol. 11098, pp. 87–110. Springer, Heidelberg, Germany, Barcelona, Spain (Sep 3–7, 2018). https://doi.org/10.1007/978-3-319-99073-6_5
8. Bowe, S., Grigg, J., Hopwood, D.: Halo: Recursive proof composition without a trusted setup. Cryptology ePrint Archive, Report 2019/1021 (2019), `https://eprint.iacr.org/2019/1021`
9. Brandt, F.: How to obtain full privacy in auctions. Int. J. Inf. Sec. **5**(4), 201–216 (2006)
10. Brickell, E.F., Chaum, D., Damgård, I., van de Graaf, J.: Gradual and verifiable release of a secret. In: Pomerance, C. (ed.) CRYPTO'87. LNCS, vol. 293, pp. 156–166. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 1988). https://doi.org/10.1007/3-540-48184-2_11
11. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy. pp. 315–334. IEEE Computer Society Press, San Francisco, CA, USA (May 21–23, 2018). https://doi.org/10.1109/SP.2018.00020
12. Cachin, C.: Efficient private bidding and auctions with an oblivious third party. In: Motiwalla, J., Tsudik, G. (eds.) ACM CCS 99. pp. 120–127. ACM Press, Singapore (Nov 1–4, 1999). https://doi.org/10.1145/319709.319726
13. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO'92. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 1993). https://doi.org/10.1007/3-540-48071-4_7
14. David, B., Gentile, L., Pourpouneh, M.: FAST: Fair auctions via secret transactions. In: Ateniese, G., Venturi, D. (eds.) ACNS 22. LNCS, vol. 13269, pp. 727–747. Springer, Heidelberg, Germany, Rome, Italy (Jun 20–23, 2022). https://doi.org/10.1007/978-3-031-09234-3_36
15. Dreier, J., Jonker, H., Lafourcade, P.: Defining verifiability in e-auction protocols. In: Chen, K., Xie, Q., Qiu, W., Li, N., Tzeng, W. (eds.) 8th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '13, Hangzhou, China - May 08 - 10, 2013. pp. 547–552. ACM (2013)

16. Dreier, J., Lafourcade, P., Lakhnech, Y.: Formal verification of e-auction protocols. In: Basin, D.A., Mitchell, J.C. (eds.) Principles of Security and Trust - Second International Conference, POST 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7796, pp. 247–266. Springer (2013)
17. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO'84. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 1984)
18. Franklin, M.K., Reiter, M.K.: The design and implementation of a secure auction service. IEEE Trans. Software Eng. **22**(5), 302–312 (1996)
19. Galal, H.S., Youssef, A.M.: Verifiable sealed-bid auction on the Ethereum blockchain. In: Zohar, A., Eyal, I., Teague, V., Clark, J., Bracciali, A., Pintore, F., Sala, M. (eds.) FC 2018 Workshops. LNCS, vol. 10958, pp. 265–278. Springer, Heidelberg, Germany, Nieuwpoort, Curaccao (Mar 2, 2019). https://doi.org/10.1007/978-3-662-58820-8_18
20. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences **28**(2), 270–299 (1984)
21. Kothapalli, A., Setty, S., Tzialla, I.: Nova: Recursive zero-knowledge arguments from folding schemes. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part IV. LNCS, vol. 13510, pp. 359–388. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–18, 2022). https://doi.org/10.1007/978-3-031-15985-5_13
22. Lafourcade, P., Nopere, M., Picot, J., Pizzuti, D., Roudeix, E.: Security analysis of auctionity: A blockchain based e-auction. In: Benzekri, A., Barbeau, M., Gong, G., Laborde, R., García-Alfaro, J. (eds.) Foundations and Practice of Security - 12th International Symposium, FPS 2019, Toulouse, France, November 5-7, 2019, Revised Selected Papers. Lecture Notes in Computer Science, vol. 12056, pp. 290–307. Springer (2019). https://doi.org/10.1007/978-3-030-45371-8_18, `https://doi.org/10.1007/978-3-030-45371-8\_18`
23. Li, H., Xue, W.: A blockchain-based sealed-bid e-auction scheme with smart contract and zero-knowledge proof. Secur. Commun. Networks **2021**, 5523394:1–5523394:10 (2021)
24. Lipmaa, H., Asokan, N., Niemi, V.: Secure Vickrey auctions without threshold trust. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 87–101. Springer, Heidelberg, Germany, Southampton, Bermuda (Mar 11–14, 2003)
25. Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: Feldman, S.I., Wellman, M.P. (eds.) Proceedings of the First ACM Conference on Electronic Commerce (EC-99), Denver, CO, USA, November 3-5, 1999. pp. 129–139. ACM (1999). https://doi.org/10.1145/336992.337028, `https://doi.org/10.1145/336992.337028`
26. Nurmi, H., Salomaa, A.: Cryptographic protocols for Vickrey auctions. Group Decision and Negotiation **2**(4), 363–373 (1993)
27. Schlegel, J.C., Mamageishvili, A.: On-chain auctions with deposits. CoRR **abs/2103.16681** (2021), `https://arxiv.org/abs/2103.16681`
28. Shi, Z., de Laat, C., Grosso, P., Zhao, Z.: Integration of blockchain and auction models: A survey, some applications, and challenges. IEEE Communications Surveys and Tutorials **25**(1), 497–537 (2023). https://doi.org/10.1109/COMST.2022.3222403
29. Vickrey, W.: Counterspeculation, auctions, and competitive sealed tenders. In: Journal of finance. vol. 16, pp. 8–37, March 1961

A Bypassing the comparison proof of [19]

Let x_w the winning bid, and let x_i any other bid. The authors of [19] observe that if $x_w, x_i, \Delta_i \mod q \in [0, q/2]$, where $\Delta_i := x_w - x_i$, then $x_w > x_i$. Thus, since the winning bid x_w is known to everyone, the comparison proof for "$x_w > x_i$" can be accomplished with two range proofs: one to prove "$x_i \in [0, q/2)$", and another to prove "$\Delta_i \mod q \in [0, q/2)$".

The range proof used in [19] is the Σ-protocol due to Brickell et. al. [10]. See Figure 10 for a description of the protocol. An (honest) prover, that knows an integer $x \in [0, B)$, convinces the verifier that $x \in [-B, 2B)$. Note the gap in the ranges, which is also observed in [19]. Since bids cannot be negative (at least if aim to be the highest bid), the authors assumed that the interval membership was reduced to $[0, 2B)$ and set the upper bound to $B = q/4$ accordingly. Our observation is that the difference $\Delta_i = x_w - x_i$ (without modulus reduction) can be negative. Below we show how a malicious prover (auctioneer) can generate a valid proof for $\Delta_i \in [-B, 0)$ that convinces the verifier that $\Delta_i \in [-B, 2B]$. In other words, how the malicious auctioneer can generate a valid comparison proof that does not attest for the veracity of "$x_w > x_i$".

The attack. Suppose the difference between the highest bid x_1, and the second-highest bid x_2, is less than $B < q/4$ and that a malicious auctioneer announces the winning bid to be $x_w := x_2$. Thus $\Delta_1 = x_w - x_1 \in [-B, 0]$. The malicious prover (auctioneer), instead of following the steps of Figure 10 proceeds slighlty different:

- In step 1 (commit), the only difference is that the auctioneer chooses $w_1 \in [-\Delta, B]$ (instead of $w_1 \in [0, B]$).
- In step 3 (response), if challenge bit is $e = 1$ the auctioneer always sends $m = \Delta + w_1$, (and the other response n).

To see why verification passes, it is enough to observe that $w_1 \in [0, B]$ (because $-\Delta$ is positive), so the verifier checks will pass in case the challenge bit is $e = 0$. Also, $m \in [0, B]$ which follows from assuming $\Delta \in [-B, 0]$ and $w_1 \in [-\Delta, B]$, so if $e = 1$ the verifier checks will also pass.

Public input Pedersen commitment $C = G^x H^o$, and upper bound B.
Private input Integer $x \in [0, B)$, and opening o.

1. **Commit.** The prover (auctioneer) picks $w_1 \in [0, B]$, sets $w_2 = w_1 - B$ and commits to w_1, w_2 with Pedersen using openings r_1, r_2. It sends commitments $W_1 = G^{w_1} H^{r_1}, W_2 = G^{w_2} H^{r_2}$ to the verifier.
2. **Challenge.** The verifier picks a random challenge bit e and sends it to the prover.
3. **Response.** The prover sends one of the following responses to the verifier, based on the challenge bit e:
   - If $e = 0$ the prover sends w_1, w_2, r_1, r_2 to the verifier. The verifier checks $w_1 \in [0, B]$, $w_2 = B - w_1$, and $W_i = G^{w_i} H^{r_i}$ for $i \in \{1, 2\}$.
   - Else $e = 1$, and the prover sends $m = x + w_j$, opening $n = o + r_j$, and index j such that $m \in [0, B]$. The verifier checks $CW_j = G^m H^n$, and $m \in [0, B]$.

Fig. 10: Range proof from [10] used in the comparison proof of [19]. It convinces the verifier that $x \in [-B, 2B]$