Cryptobazaar: Private Sealed-bid Auctions at Scale

Andrija Novakovic
Geometry Research
andrija@geometry.dev

Alireza Kavousi
University College London
a.kavousi@cs.ucl.ac.uk

Kobi Gurkan
Geometry Research
kobi@geometry.dev

Philipp Jovanovic
University College London
p.jovanovic@ucl.ac.uk

Abstract

This work introduces Cryptobazaar, a novel scalable, private, and decentralized sealed-bid auction protocol. In particular, our protocol protects the privacy of losing bidders by preserving the confidentiality of their bids while ensuring public verifiability of the outcome and relying only on a single untrusted auctioneer for coordination. At its core, Cryptobazaar combines an efficient distributed protocol to compute the logical-OR for a list of unary-encoded bids with various novel zero-knowledge succinct arguments of knowledge that may be of independent interest. We further present variants of our protocol that can be used for efficient first-, second-, and more generally $(p+1)$st-price as well as sequential first-price auctions. Finally, the performance evaluation of our Cryptobazaar implementation shows that it is highly practical. For example, a single run of an auction with 128 bidders and a price range of 1024 values terminates in less than 0.5 sec and requires each bidder to send and receive only about 32 KB of data.

1 Introduction

An auction is a sales process in which potential buyers aim to acquire goods or services by placing competitive bids. Auctions thus facilitate transactions by enforcing a specific set of rules regarding the resource allocation of a group of bidders. An auctioneer usually coordinates an auction process and may also be a seller of goods or services. There are various different types of auctions with the four main ones being first-price sealed-bid, second-price sealed-bid (Vickrey), ascending open-bid (English), and descending open-bid (Dutch). Throughout history auctions have been used for countless different purposes. They also play a crucial role in many digital systems such as for selling and buying of advertisements [54], for allocating block space and determining the transaction order to mitigate maximal extractable value (MEV) in cryptocurrencies [22, 46], or for renting out hardware to run large-scale computations or to store data [36], and more.

A digital auction system should provide various essential properties, including: (1) *privacy* for (losing) bidders to prevent disclosure of their bid preferences [36] which is particularly crucial to ensure fairness in iterative auctions, as any leakage may allow bidders to adapt their strategies in subsequent runs [1, 22, 31]; (2) *verifiability* to ensure that third parties (auditors) can verify the outcome and penalize misbehaving participants [31]; (3) *trust minimization* to mitigate the influence that a potentially malicious auctioneer could have on the auction's outcome [50]; and (4) *scalability* to ensure that the auction protocol can meet the requirements of the application (in terms of the number of bidders or price ranges) in which it is used and not be a performance bottleneck (in terms of computational and communication overheads) [36].

There have been many attempts to design sealed-bid auction systems with the above properties but they usually fall short in one way or another: In the commonly used commit-reveal approach, bidders first commit to their bids and open them later after all bids have been submitted [51]. This approach has incentive misalignments though given that bids may be selectively revealed and it is usually difficult to protect the privacy of losing bidders. There have been recent attempts to address the incentive problem through time-based cryptography [27, 48, 56] but none have managed to address the privacy issue as well so far. Other approaches rely on heavy cryptographic machinery such as generic secure multi-party computation (MPC) [8] or fully homomorphic encryption (FHE) [32] to address the privacy challenges which, however, severely impacts scalability making these protocols not practically useful or restricting them to small scale settings. In a recent attempt, Addax [57] achieves both privacy and scalability but only under a weakened threat model that requires non-colluding assumption among (two or more) auctioneers.

In this work, we propose Cryptobazaar, a novel private auction protocol which improves over the state-of-the-art by addressing the previously identified challenges. In particular, Cryptobazaar provides (1) *privacy* for (losing) bidders/bids by only revealing the the winning bid and the sale price; (2) *verifiability* to ensure that all protocol steps can be *publicly* verified by anyone (*e.g.,* auditors) to flag any misbehavior; (3) *trust minimization* to support auction execution in peer-to-peer mode while also allowing a single untrusted auctioneer as coordinator for additional efficiency; (4) *scalability* to support a large number of bidders and price ranges while having low overheads in terms of computation and bandwidth consumption for bidders and auctioneer; and (5) *versatility* to support first-, second-, and more general $(p+1)$-price auctions (*i.e.,* uniform auction) as well as an iterative mode with only a slight adjustment of the main protocol.

Technical overview. In Cryptobazaar we encode bids as unary vectors enabling us to determine the results of an auction run in a privacy-preserving yet scalable way via a distributed Boolean-OR over all bids using the *anonymous veto protocol* [30]. To ensure the well-formedness of the submitted values while maintaining privacy, we design and use various new efficient zero-knowledge proofs. These include, for example, arguments for proving correctness of unary encodings which could be also useful for other applications like voting. We further show how to link Pederson commitments to univariate sumcheck and how to replace an inner-product argument with a univariate sumcheck to improve prover efficiency. These newly developed techniques on *inner-product arguments* [9, 13], *log-derivatives* [29], and *univariate sumcheck* [6] may be of independent interest. We break possible ties via public randomness [34] and efficient zero-knowledge set membership proofs preserving the privacy of the winner. Moreover, we show how to extend the base version of Cryptobazaar to second- and more generally $(p+1)$st-price auctions efficiently without re-running the protocol which is a common practice for a private Vickery auction [39]. Finally, we propose a variant to run secure iterative auctions using setup re-randomization techniques, making Cryptobazaar particularly appealing for practical use cases. As a concrete application, Cryptobazaar could be an alternative for the auction protocol used in the Ethereum block building process [22] demanding the auction to terminate in a fraction of its 12-second block time window.

In summary, we make the following contributions:

- We propose Cryptobazaar, a scalable private sealed-bid auction protocol that supports first- and second-price auctions and relies only on a untrusted auctioneer for coordination.
- We co-design various novel (public coin) zero-knowledge succinct arguments of knowledge to ensure the well-formedness of the different protocol steps.
- We present two extensions to Cryptobazaar showing how to support $(p+1)$-price auctions and iterative auctions efficiently without having to re-run the full protocol.
- We demonstrate the practicality of our system by evaluating an open-source implementation of Cryptobazaar in Rust. Concretely, a run of the auction with 128 bidders and a price range of 1024 values terminates in less than 0.5 seconds and requires each bidder to communicate only about 32 KB of data.

The remainder of the paper is organized as follows: Section 2 provides the required background, Section 3 introduces Cryptobazaar and the protocol details, Section 4 discusses the zero-knowledge proof techniques that we developed, Section 5 provides various extensions of Cryptobazaar, Section 6 presents the evaluation results for our implementation, and Section 7 gives an overview on related work.

2 Background

2.1 Notation

We write $x \overset{\$}{\leftarrow} S$ to denote uniform sampling of element x from finite set S and $[n]$ for the set of integers $\{1,\dots,n\}$. We denote by $\mathbb{G}$ a cyclic group of prime order p and by $\mathbb{F}$ a finite (scalar) field. We write group operations additively. We indicate group elements by uppercase letters $G \in \mathbb{G}$, and scalars by lowercase letters $a \in \mathbb{F}$. Given a scalar $a \in \mathbb{F}$ and group element $G \in \mathbb{G}$, we denote scalar multiplication by $a \cdot G \in \mathbb{G}$. We express vectors of elements in boldface, *e.g.,* $\boldsymbol{a} = (a_1,\dots,a_n) \in \mathbb{F}^n$. We denote the inner product of two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}^n$ as $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \sum_{i=1}^{n} a_i b_i$ and the entry-wise multiplication of two vectors as $\boldsymbol{a} \circ \boldsymbol{b} = (a_1 \cdot b_1, \dots, a_n \cdot b_n)$. To denote the scalar multiplication of scalar $a \in \mathbb{F}$ with every element of a vector of group elements $\boldsymbol{G} \in \mathbb{G}^n$, we write $a \cdot \boldsymbol{G} = (a \cdot G_1, \dots, a \cdot G_n)$. Given $1 \le l \le n$ and vector $\boldsymbol{a} \in \mathbb{F}^n$, we denote the slice of the first l elements of $\boldsymbol{a}$ by $\boldsymbol{a}_{[:l]} = (a_1,\dots,a_l) \in \mathbb{F}^l$ and the slice of the last $n-l$ elements of $\boldsymbol{a}$ by $\boldsymbol{a}_{[l:]} = (a_{l+1},\dots,a_n) \in \mathbb{F}^{n-l}$. A non-empty subset $\mathbb{H} \subseteq \mathbb{F}$ is said to be a multiplicative subgroup of field $\mathbb{F}$ if $\mathbb{H}$ is closed under multiplication and inverses. Given a non-empty subset $\mathbb{H} \subseteq \mathbb{F}$ we denote by $z_{\mathbb{H}}(X) = \prod_{w \in \mathbb{H}} (X - w)$ the vanishing polynomial of $\mathbb{H}$. Capitalized bold font $\mathbf{A} \in \mathbb{F}^{n \times m}$ denotes a matrix, with n rows and m columns. Blinding factors are denoted by Greek letters. We denote the security parameter by $\lambda \in \mathbb{N}$ and implicitly assume for a given n that $n = \text{poly}(\lambda)$.

2.2 Bilinear Groups

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of prime order p with generators G and H, respectively. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear map with the following properties.

- **Bilinearity.** For all $U \in \mathbb{G}_1$ and $V \in \mathbb{G}_2$ and $a, b \in \mathbb{F}$, we have $e(aU, bV) = e(U,V)^{ab}$.
- **Non-degeneracy.** It holds that $e(G,H) \neq 1$.

We call $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, G, H, e)$ a *bilinear group* if e is efficiently computable. For ease of notation, we define $[x]_1 = xG \in \mathbb{G}_1$, and $[x]_2 = xH \in \mathbb{G}_2$. We sometimes also write $[1]_1$ and $[1]_2$ for the generators G of $\mathbb{G}_1$ and H of $\mathbb{G}_2$, respectively.

2.3 Commitment Schemes

A commitment scheme allows a sender to commit to a secret value and open it later in a verifiable way such that a receiver can check whether the revealed value is consistent with the committed one. A commitment scheme has two main security properties, namely (1) *binding*, meaning that the commitment cannot be opened to two different values, and (2) *hiding*, meaning that the commitment should not reveal any information about the committed secret value. The *Pedersen commitment scheme* [42] is an example of a widely-used commitment scheme that is perfectly hiding and computationally binding in which a sender can commit to a secret value $x \in \mathbb{F}$ using generators $G, H \in \mathbb{G}$ and randomness $r \in \mathbb{F}$ via $C = xG + rH \in \mathbb{G}$. *Vector commitments* [14] are an extension allowing a sender to commit to a set of values and open them individually later on with two popular examples being *Merkle trees* and *Pedersen vector commitments*. Finally, a *polynomial commitment scheme (PCS)* [33] is a variant of a vector commitment scheme that enables the sender to commit to a polynomial such that the receiver can confirm claimed evaluations of the committed polynomial later. In more detail:

Definition 1 (Polynomial Commitment Scheme (PCS))**.** A polynomial commitment scheme PCS is defined by the following algorithms:

- $\mathsf{SP} \leftarrow \mathsf{PCS.Setup}(1^\lambda, d)$: Takes as input a security parameter λ and an integer d, and outputs system parameters SP to commit to a polynomial of degree $\leq d$.
- $C \leftarrow \mathsf{PCS.Commit}(\mathsf{SP}, \phi(\cdot))$: Takes as input the system parameters SP and a polynomial $\phi(\cdot)$, and outputs a commitment C to $\phi(\cdot)$.
- $(\pi, \phi(w)) \leftarrow \mathsf{PCS.Open}(\mathsf{SP}, \phi(\cdot), w)$: Takes as input the system parameters SP, the polynomial $\phi(\cdot)$, and a point w, and outputs the polynomial evaluation $\phi(w)$ and an evaluation proof π.
- $1/0 \leftarrow \mathsf{PCS.Verify}(\mathsf{SP}, C, w, \phi(w), \pi)$: Takes as input the system parameter SP, the commitment C, a point w, the evaluation $\phi(w)$, and the proof π, and verifies that $\phi(w)$ is indeed the evaluation of polynomial ϕ at value w in commitment C using proof π.

A PCS furthermore satisfies the following security properties:

- **Correctness.** Given $C \leftarrow \mathsf{PCS.Commit}(\mathsf{SP}, \phi(\cdot))$ and $(\pi, \phi(w)) \leftarrow \mathsf{PCS.Open}(\mathsf{SP}, \phi(\cdot), w)$, then $\mathsf{PCS.Verify}(\mathsf{SP}, C, w, \phi(w), \pi) = 1$ with probability 1.
- **Polynomial Binding.** An adversary should not be able to produce two different polynomials $\phi(\cdot)$ and $\phi'(\cdot)$ such that $\mathsf{PCS.Commit}(\mathsf{SP}, \phi(\cdot)) = \mathsf{PCS.Commit}(\mathsf{SP}, \phi'(\cdot))$, except with negligible probability.
- **Evaluation Binding.** An adversary should not be able to produce values $\{C, (w, \phi(w), \pi), (w, \phi'(w), \pi')\}$ with $\mathsf{PCS.Verify}(\mathsf{SP}, C, w, \phi(w), \pi) = 1$, $\mathsf{PCS.Verify}(\mathsf{SP}, C, w, \phi'(w), \pi') = 1$, and $\phi(w) \neq \phi'(w)$, except with negligible probability.

In our protocols we use the popular KZG PCS [33]:

- $\mathsf{SP} \leftarrow \mathsf{KZG.Setup}(1^\lambda, d)$: Outputs a bilinear pairing group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, G, H, e)$, and system parameters $\mathsf{SP} = [\tau^i]_1 = (G, \tau G, \ldots, \tau^d G)$ for trapdoor $\tau \in \mathbb{F}$ which is generated by a (distributed) trusted authority.
- $C \leftarrow \mathsf{KZG.Commit}(\mathsf{SP}, \phi(\cdot))$: Outputs the commitment $C = [\phi(\tau)]_1 = \sum_{i=0}^{d} \phi_i [\tau^i]_1$.
- $(\pi, \phi(w)) \leftarrow \mathsf{KZG.Open}(\mathsf{SP}, \phi(\cdot), w)$: Computes the quotient polynomial $q(X) = (\phi(X) - \phi(w)) \cdot (X - w)^{-1}$ and outputs evaluation $\phi(w)$ and proof $\pi = [q(\tau)]_1$.
- $1/0 \leftarrow \mathsf{KZG.Verify}(\mathsf{SP}, C, w, \phi(w), \pi)$: Outputs 1 if $e(\pi, [\tau - w]_2) = e(C - [\phi(w)]_1, [1]_2)$ holds and 0 otherwise.

The basic version of KZG is not hiding since it is deterministic. However, in some situations having hiding is desirable. This means that no PPT adversary should learn anything about an unqueried evaluation point w' given information of up to d evaluations. By masking commitment and proof with randomizers, one can turn KZG into *blinded KZG* [55] and achieve hiding. KZG requires a setup phase to compute system parameters SP that include a trapdoor τ. To ensure that no individual party knows τ, which would enable them to undermine the security of KZG, one can use a distributed protocol to compute SP ensuring security as long as there is a single honest contribution [40]. Alternatively, one could use a PCS that does not require a trusted setup at the cost of a performance trade-off [5].

2.4 Anonymous Veto

In Cryptobazaar we use the *anonymous veto (AV)* protocol [30] as one of the main building blocks. AV is an efficient two-round protocol for computing the logical-OR function over a set of input bits without revealing any information about the individual ones. AV is run over a cyclic group $\mathbb{G}$ of prime order p with a generator G in which the Decision Diffie-Hellman (DDH) problem is hard. The AV protocol works as follows:

- Each party i samples a random value $x_i \in \mathbb{F}$, and broadcasts $x_i G$. After seeing messages from all other parties, i computes $Y_i = \sum_{j=1}^{i-1} x_j G - \sum_{j=i+1}^{n} x_j G$.
- After obtaining Y_i, each party i computes a value R_i as either $R_i = r_i Y_i$ if they veto or $R_i = x_i Y_i$ if they do not veto where $r_i \xleftarrow{\$} \mathbb{F}$, and sends R_i to all of the other participants.

The output of the AV protocol is obtained by having each party compute $R = \sum_{i=1}^{n} R_i$. If no one vetoes then $R = 0$, otherwise if at least one party vetoes then $R \neq 0$. This follows from the definition of Y_i that implies $\sum_i x_i Y_i = 0$ (see Proposition 1, [30]). To illustrate this equality more intuitively, let X be the vector of elements obtained after the first AV round, *i.e.,*

$X_i = x_i G$, then we define matrix $\mathbf{M}$ as

$$\mathbf{M} = \begin{bmatrix} 0 & -1 & -1 & \cdots & -1 \\ 1 & 0 & -1 & \cdots & -1 \\ 1 & 1 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 & -1 \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix}.$$

Now it holds $\langle X, \mathbf{M} \cdot X^T \rangle = 0$. The AV protocol preserves the privacy of a participating party as long as not all the other parties collude. The security of this scheme stems from the inability of an attacker to distinguish between $r_i Y_i$ and $x_i Y_i$ under the DDH assumption. Later we show how the structure of $\mathbf{M}$ can be used to efficiently delegate the computation of Y_i for $i \in [n]$ to a single untrusted party.

2.5 Zero-knowledge Argument of Knowledge

In Cryptobazaar we deploy *zero-knowledge succinct non-interactive arguments of knowledge (zkSNARKs)* [41] to enforce honest behavior of the parties and ensure public verifiability. These are defined as follows:

Definition 2 (Argument System)**.** A (non-interactive) argument system for an efficiently decidable binary relation $\mathcal{R}$ for an NP language $L_{\mathcal{R}}$ is a tuple of probabilistic polynomial algorithms $\mathsf{AS} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Simulate})$ such that:

- $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{AS.Setup}(1^\lambda, \mathcal{R})$: The setup algorithm takes as input a security parameter λ and a relation $\mathcal{R}$ and outputs a common reference string crs and a trapdoor td.
- $\pi \leftarrow \mathsf{AS.Prove}(\mathsf{crs}, u, w)$: The prover algorithm takes as input the crs, a statement u and a witness w and outputs an argument π.
- $0/1 \leftarrow \mathsf{AS.Verify}(\mathsf{crs}, u, \pi)$: The verifier algorithm takes as input the crs, a statement u, and an argument π and outputs 1 (accept) if the proof was accepted and 0 otherwise.
- $\pi \leftarrow \mathsf{AS.Simulate}(\mathsf{crs}, \mathsf{td}, u)$: The simulator takes as input the crs, a simulation trapdoor td, and a statement u and outputs an argument π.

For an argument system to be a zkSNARK, we require certain security properties to protect the prover from witness leakage and the verifier from a forged proof. Informally, an argument system should have *completeness*, where an honest prover always convinces the verifier about a true statement. It also requires *knowledge soundness* demanding the prover to indeed know the witness whenever it generates a valid proof. Lastly, an argument system should be *zero-knowledge* and does not leak any information besides the truth of the statement. We provide the formal definitions in Appendix A.1. An argument system is *public coin* if all the messages sent form the verifier have uniform distribution and are independent from the ones received from the prover. This then allows to make the protocol non-interactive using Fiat-Shamir [21].

Note that all our arguments throughout the paper will be public coin and thus can be made non-interactive. We prove the security of our arguments in the *Algebraic Group Model (AGM)* [25]. There are different ways to build zkSNARKs concretely. We follow the approach in *PLONK* [26] by taking polynomial *interactive oracle proofs* (IOP) [7], combining it with a polynomial commitment scheme to obtain succinct interactive arguments, and applying the Fiat-Shamir transform to make it non-interactive. Below we recall the basics for some of the techniques that we use in our zkSNARKs.

2.5.1 Inner-product Arguments (IPA)

An inner-product argument [9] is an efficient argument system for the following relation:

$$\mathcal{R} = \{(\boldsymbol{G}, \boldsymbol{H} \in \mathbb{G}^n, P \in \mathbb{G}, c \in \mathbb{F}_p; \boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}_p^n) : \begin{array}{l} P = \langle \boldsymbol{a}, \boldsymbol{G} \rangle + \langle \boldsymbol{b}, \boldsymbol{H} \rangle, \\ c = \langle \boldsymbol{a}, \boldsymbol{b} \rangle \} \end{array}$$

An IPA convinces the verifier that the prover knows the openings of two Pedersen vector commitments satisfying a given inner product relation. Bunz et al. [12] proposed an improved IPA where the inner-product value c is hidden as part of the vector commitment P and with logarithmic proof size in the vector dimension n. IPA can be generalized to capture other types of inner products, including ones that work with bilinear pairings [13].

2.5.2 Univariate Sumcheck Protocol

The univariate sumcheck protocol [6] relates the sum of any (low-degree) polynomial over a multiplicative subgroup $\mathbb{H}$ of field $\mathbb{F}$ to the polynomial's evaluation at a single point. The following lemma offers a polynomial IOP that we use for constructing some of our arguments.

Lemma 1 (Univariate Sumcheck for Subgroups)**.** Given a multiplicative subgroup $\mathbb{H}$ of field $\mathbb{F}$, a polynomial $f(X)$ sums to σ over $\mathbb{H}$ if and only if $f(X)$ can be written as $Xg(X) + h(X)z_{\mathbb{H}}(X) + \sigma/|\mathbb{H}|$.

Interactive argument systems exploit the following basic property of polynomials, which is commonly known as the Schwartz-Zippel lemma [58].

Lemma 2 (Schwartz-Zippel)**.** Let $\mathbb{F}$ be any field, and let $g : \mathbb{F}^m \to \mathbb{F}$ be a nonzero polynomial of total degree at most d. Then, on any finite set $S \subseteq \mathbb{F}$ and for any $\boldsymbol{x} \leftarrow S^m$, $\Pr[g(\boldsymbol{x}) = 0] \leq \frac{d}{|S|}$.

An implication of the Schwartz-Zippel lemma is that for any two distinct (univariate) polynomials ϕ_1 and ϕ_2 of total degree at most d over $\mathbb{F}$, they agree (*i.e.,* $\phi_1(x) = \phi_2(x)$) mostly at $d/|\mathbb{F}|$ fraction of inputs. So, one could verify that a polynomial relation holds with overwhelming probability by evaluating it at a random point.

2.5.3 Logarithmic Derivatives

As in basic calculus, the logarithmic derivative of a polynomial $\phi(X)$ over a field $\mathbb{F}$ is defined as $\phi'(X)\cdot\phi(X)^{-1}$. Further, the logarithmic derivative of a product function $\phi(X) = \prod_{i=1}^{n}(X+z_i)$, where $z_i \in \mathbb{F}$, is equal to the sum of its reciprocals, *i.e.,* $\phi'(X)\cdot\phi(X)^{-1} = \sum_{i=1}^{n}(X+z_i)^{-1}$. Further if two normalized polynomials have the same logarithmic derivative, they are equal, as stated by the following lemma [29].

Lemma 3. Let $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}_p^n$ with $p > n$. Then, $\prod_{i=1}^{n}(X+a_i) = \prod_{i=1}^{n}(X+b_i)$ if and only if $\sum_{i=1}^{n}(X+a_i)^{-1} = \sum_{i=1}^{n}(X+b_i)^{-1}$.

Further, Haboeck [29] shows that it is possible to obtain the unique fractional decomposition of the logarithmic derivative of a product function $\phi(X) = \prod_{i=1}^{n}(X+z_i)$ via $\phi'(X)\cdot\phi(X)^{-1} = \sum_{z\in\mathbb{F}} m(z)\cdot(X+z)^{-1}$ where $m(z) \in [n]$ is the multiplicity of the value z in $z_1,\ldots,z_n$.

3 Cryptobazaar

In this section we present Cryptobazaar, our private scalable sealed-bid auction protocol. In Section 3.1 we discuss system and threat models, in Section 3.2 we present the design goals, and in Section 3.3 we discuss the protocol details.

3.1 System and Threat Models

In Cryptobazaar we assume that there are m bidders, an untrusted auctioneer, and one or more auditors with access to an append-only public log (*e.g.,* a public ledger) and a price range of n values. Each bidder communicates with the auctioneer through a public authenticated channel. We do not assume any point-to-point channel (and thus interaction) between bidders as they might have no priori information about the number of protocol participants or their identities. Our communication pattern resembles a star topology with the auctioneer being the coordinator. Note that Cryptobazaar can also work in a peer-to-peer setting without an auctioneer but we prefer the auctioneer-based mode henceforth due to its lower bandwidth consumption and since they are usually present in practice [57].

We model bidders and auctioneer as *covert adversaries* [2] who may arbitrarily deviate from the correct execution of the protocol as long as they are not detected. If they are caught, they might be subject to penalties (*e.g.,* financial or reputational). The auditor gets involved at the end of the auction to validate the outcome using the information posted on the public log. However, no information should be leaked beyond what can already be inferred from the auction including the (second) highest bid. We assume that authentication between bidders and auctioneer and the penalizing of misbehaving parties is handled out of band.

3.2 Design Goals

We aim to achieve the following goals in designing Cryptobazaar:

- **Privacy:** The protocol only discloses the (second) highest bid and protects the privacy of (losing) bidders.
- **Verifiability:** If any participant misbehaves during the protocol execution, verification fails and the cheating party is detected with overwhelming probability. All protocol steps are publicly verifiable and auditors could verify the protocol run using the data posted on a public log.
- **Trust minimization:** The protocol runs in a decentralization fashion where bidders do not necessarily know each other and there is no trust on the auctioneer as a coordinator.
- **Scalability**: The protocol supports a large number of bidders and price ranges while having low overheads in terms of computation and bandwidth consumption for bidders and auctioneer.

3.3 Protocol Description

Figure 1 presents an overview of Cryptobazaar which consists of the two phases *preprocessing* and *bidding*. Below we focus on the core protocol and present the necessary validity proofs only abstractly. We defer the discussion of the proofs' technical details to Section 4. Note that these proofs are computed asynchronously and sent to the public log to be verified later.

Preprocessing phase. Each bidder initially samples two vectors $\boldsymbol{x}, \boldsymbol{r} \in \mathbb{F}^n$, and computes $\boldsymbol{X} = \boldsymbol{x}\cdot G \in \mathbb{G}^n$. Suppose all the input values (*i.e.,* bids) are integers in the range $[0, n]$. Each bidder encodes their bid as a unary vector $\boldsymbol{b} = (b_1,\ldots,b_n)$, where $b_j = 1$ if and only if $j \leq b$. For example, for $n = 5$ the bid $b = 3$ is represented as 11100. Observe that the values in $\boldsymbol{X}$ are randomly sampled and thus are independent from the bids. Furthermore, each bidder computes the polynomial commitments $\mathtt{x}, \mathtt{r}, \mathtt{b}$, constructs three validity proofs π_x, π_r, π_b, and appends $(\mathtt{x}, \mathtt{r}, \mathtt{b}, \boldsymbol{X}, \pi_x, \pi_r, \pi_b)$ to the public log.

The auctioneer then carries out n runs of the AV protocol (first round) in parallel on the values received from the bidders and computes the vectors $\boldsymbol{Y}_i$ for $i \in [m]$ as the row-vectors of a $m \times n$ matrix $\mathbf{Y} = \mathbf{M}\cdot\mathbf{X}$, where $\mathbf{M}$ is the AV matrix (Section 2.4), and $\mathbf{X}$ contains row-vectors $\boldsymbol{X}$. Finally, the auctioneer appends the row-vectors $\boldsymbol{Y}_i$ to the public log and optionally sends $\boldsymbol{Y}_i$ to bidder i for all i.

Bidding phase. After a bidder received its individual vector $\boldsymbol{Y}_i$, they compute $\boldsymbol{Z}_i = (\boldsymbol{x}_i + \boldsymbol{b}_i \circ \boldsymbol{r}_i) \circ \boldsymbol{Y}_i$. The values of random vector $\boldsymbol{r}_i$ are enabled depending on the values of the unary vector $\boldsymbol{b}_i$. So, for each $j \in [n]$ we have either $\boldsymbol{Z}_{ij} = (\boldsymbol{x}_{ij} + \boldsymbol{r}_{ij})\boldsymbol{Y}_{ij}$ if $\boldsymbol{b}_{ij} = 1$ or $\boldsymbol{Z}_{ij} = \boldsymbol{x}_{ij}\boldsymbol{Y}_{ij}$ if $\boldsymbol{b}_{ij} = 0$. Each bidder furthermore computes a validity proof π_{Z_i} showing the well-formedness of the vector $\boldsymbol{Z}_i$, *i.e.,* its consistency with their committed vectors $\mathtt{x}_i, \mathtt{r}_i, \mathtt{b}_i$ and vector $\boldsymbol{Y}_i$. Then, the auctioneer completes the AV protocols and determines the outcome by computing

Preprocessing phase

Each bidder $i \in [m]$ executes the following steps:

1. Sample random non-zero vectors $\boldsymbol{x}_i, \boldsymbol{r}_i \in \mathbb{F}^n$.
2. Create polynomial commitments $\mathtt{x}_i$ of $\boldsymbol{x}_i$ and $\mathtt{r}_i$ of $\boldsymbol{r}_i$.
3. Compute vector $\boldsymbol{X}_i$ such that $\boldsymbol{X}_i = \boldsymbol{x}_i \cdot G$.
4. Create a proof π_{x_i} showing that $\mathtt{x}_i$ and $\boldsymbol{X}_i$ both encode the same vector $\boldsymbol{x}_i$.
5. Create a proof π_{r_i} showing that $\boldsymbol{r}_j \neq 0$ for each $j \in [n]$.
6. Decide on bid b_i and compute its unary encoding $\boldsymbol{b}_i$.
7. Compute a (blinded) polynomial commitment $\mathtt{b}_i$ of $\boldsymbol{b}_i$.
8. Create a proof π_{b_i} that $\boldsymbol{b}_i$ is a valid unary encoding.
9. Append $(\mathtt{x}_i, \mathtt{r}_i, \mathtt{b}_i, \boldsymbol{X}_i, \pi_{x_i}, \pi_{r_i}, \pi_{b_i})$ to the public log.

Once all bidders are done, the auctioneer executes the following steps:

10. Load row-vectors $\boldsymbol{X}_i$ to compute $m \times n$ matrix $\mathbf{Y} = \mathbf{M} \cdot \mathbf{X}$ where $\mathbf{M}$ is the matrix as specified in Section 2.4.
11. Append row-vector $\boldsymbol{Y}_i$ to the public log for each $i \in [m]$.

Bidding phase

Each bidder $i \in [m]$ executes the following steps:

1. Load row-vector $\boldsymbol{Y}_i$ and compute vector $\boldsymbol{Z}_i$ such that $\boldsymbol{Z}_i = (\boldsymbol{x}_i + \boldsymbol{b}_i \circ \boldsymbol{r}_i) \circ \boldsymbol{Y}_i$.
2. Create a proof π_{Z_i} showing that $\boldsymbol{Z}_i$ is of the above form.
3. Append $(\mathtt{b}_i, \boldsymbol{Z}_i, \pi_{b_i}, \pi_{Z_i})$ to the public log.

Once all bidders are done, the auctioneer executes the following steps:

4. Compute vector $\boldsymbol{R}$ such that $\boldsymbol{R}_j = \sum_{i=1}^{m} (\boldsymbol{Z}_i)_j$ for $j \in [n]$.
5. Append $\boldsymbol{R}$ to the public log.

The highest bid is defined by the highest index w with $\boldsymbol{R}_w \neq 0$.

6. The candidate winner constructs a proof π_w to demonstrate their eligibility.

Figure 1: The Cryptobazaar protocol

$\boldsymbol{R} = \sum_{i=1}^{m} (\boldsymbol{Z}_i)$. The index w of the highest non-zero value in $\boldsymbol{R}$ indicates the highest bid.

To find the winner in a first-price auction, we simply determine the highest bid and the corresponding bidder wins. After the winning price has been announced, a candidate winner can claim their win by providing a proof π_w showing that they have indeed bid one at position/index w. Given that the AV computes a logical-OR across all bids and thus only distinguishes whether there is at least one bid for a certain value or not, there might actually be multiple bids at the same price. In this case, the auctioneer can use public randomness [34] to choose among the set of candidate winners.

For a first-price auction, we are done at this point. For a second-price auction, we need to run the protocol again after excluding the current winner to compute the second-highest price as the sale price. In Section 5, we propose a variant that supports second-price and more generally $(p+1)$st-price auction [35] without having to re-run the protocol.

Theorem 1. The Cryptobazaar protocol (Figure 1) achieves completeness, privacy, soundness, and public verifiability.

We provide the proof to Theorem 1 in Appendix B.1.

Reducing computational overheads. The original AV protocol has linear $O(n)$ communication cost (assuming broadcast) and quadratic $O(n^2)$ computational cost for each participant. However, it is possible to reduce the computational overhead to determine the values $\boldsymbol{Y}_i$ to $O(n)$ group operations from the naive computation $\mathbf{M} \cdot \mathbf{X}$ requiring $O(n^2)$ group operations. We observe that by utilizing the relationship between consecutive rows of $\mathbf{M}$, we have $\boldsymbol{Y}_{i+1} = \boldsymbol{Y}_i + \boldsymbol{X}_i + \boldsymbol{X}_{i+1}$. Note that $\boldsymbol{Y}_0$ can be computed by a single multiscalar multiplication (MSM) [19] of size n between the first row of $\mathbf{M}$ and $\boldsymbol{X}$. In total, this leads to $O(n)$ group operations. To verify the computation, anyone with access to the transcript can simply run the same computation in $O(n)$ group operations and compare the outputs. Verifying that vectors $\boldsymbol{X}, \boldsymbol{Y}$ and $\boldsymbol{R}$ are valid requires more elaboration which we provide in Section 4.

4 Validity Proofs

In this section we present the technical details of the validity proofs used in Cryptobazaar. In Section 4.1 we discuss the preprocessing phase proofs π_{x_i} (Step 4), π_{r_i} (Step 5), and π_{b_i} (Step 8) and in Section 4.2 we discuss the bidding phase proofs π_{Z_i} (Step 2) and π_w (Step 6).

4.1 Preprocessing Phase

Proof π_{x_i} [Step 4]. The goal of this validity proof is to link a KZG polynomial commitment with a vector of elliptic curve points $\boldsymbol{X}$ to show that they both encode the same vector $\boldsymbol{x}$. More precisely, we need to develop an argument of knowledge for the following relation $\mathcal{R}_{\mathsf{pv}}$:

$$\mathcal{R}_{\mathsf{pv}} = \left\{ \left(\begin{array}{c} (\mathsf{SP}, G); \\ (\mathtt{x}, \boldsymbol{X}); \\ (\boldsymbol{x}); \end{array} \right) \middle| \begin{array}{c} f(X) = \tilde{\boldsymbol{x}} \\ \mathtt{x} = [f(\tau)]_1 \\ \boldsymbol{X}_i = \boldsymbol{x}_i G, \forall i \end{array} \right\}$$

Here $\tilde{\boldsymbol{x}}$ is the low degree extension (LDE) of $\boldsymbol{x}$. We provide the corresponding argument Π_{pv} in Figure 2 which proves that $\boldsymbol{X}$ and polynomial commitment $\mathtt{x}$ encode the same vector $\boldsymbol{x}$.

Lemma 4. The protocol Π_{pv} for relation $\mathcal{R}_{\mathsf{pv}}$, see Figure 2, satisfies completeness, soundness, and zero-knowledge.

Round 1 Verifier: Send random challenge $\gamma \in \mathbb{F}$.
Round 1 Prover:

1. Compute $q(X) = \frac{f(X)-f(\gamma)}{X-\gamma}$.
2. Send $\mathsf{q} = [q(\tau)]_1$.

Round 2 Verifier:

1. Compute $[y]_1 \leftarrow \sum_{i=1}^{n} L_i(\gamma)\mathbf{X}_i$.
2. Assert $e(\mathtt{x} - [y]_1 + \gamma\mathsf{q}, [1]_2) = e(\mathsf{q}, [\tau]_2)$.

Figure 2: Interactive zero-knowledge argument of knowledge protocol Π_{pv} for relation $\mathcal{R}_{\mathsf{pv}}$.

We provide the proof for Lemma 4 in Appendix B.2.

Proof π_{r_i} [Step 5]. The goal of this proof is to show that $\boldsymbol{r}_i \neq 0$ for the following two reasons. First, these random values are enabled according to the values of unary bid vector $\boldsymbol{b}_i$ and vice-versa. That is, setting $\boldsymbol{r}_i = 0$ essentially prevents verifying if the bidder used the same bid that they committed earlier. Second, we later show how the candidate winner can use these random values to construct a proof of eligibility. Before we proceed with describing the relation, it is helpful to state a simple but crucial technique.

Blinding. Constructing succinct arguments often requires the prover $\mathcal{P}$ to open some committed polynomial $f(X)$ at some random point γ (which is due to the Schwartz-Zippel Lemma 2 that facilitates efficient equality check of polynomials). To show that the relation $\mathcal{R}(f_1(X), f_2(X), \ldots, f_n(X))$ holds over subgroup $\mathbb{H}$ of field $\mathbb{F}$, one could instead show that $\mathcal{R}(f_1(h), f_2(h), \ldots, f_n(h)) = 0, \forall h \in \mathbb{H}$. This is equivalent to demonstrating the knowledge of some quotient polynomial $q(X)$ such that $\mathcal{R}(f_1(X), f_2(X), \ldots, f_n(X)) = q(X)z_H(X)$, where $z_H(X)$ is a vanishing polynomial of $\mathbb{H}$ [33]. Since sending plain evaluations may leak information, one can blind the (witness) polynomial to achieve zero-knowledge. To do so, we follow the approach used in Marlin [15] where the prover $\mathcal{P}$ samples a random polynomial $r(X)$ of degree (at least) equal to the number of openings they have to provide during the protocol and send $\hat{f}(X) = f(X) + r(X)z_H(X)$. So, $\hat{f}(X)$ does not leak information on $f(X)$ up to a certain number of openings, *i.e.,* zero-knowledge is provided up to a query bound [15]. Further, the use of vanishing polynomial makes the evaluations of $\hat{f}(X)$ be equal to those of $f(X)$ over $\mathbb{H}$ and thus it does not affect the relation $\mathcal{R}(f_1(X), f_2(X), \ldots, f_n(X))$.

Now assume that the number of positions n is a power of 2, let $\mathbb{H}$ be a multiplicative subgroup of size n, and let $z_H(X)$ be a vanishing polynomial of $\mathbb{H}$. Let $r(X)$ be a low degree extension of $\boldsymbol{r}$, then $\mathcal{P}$ samples a random masking polynomial $m(X)$ and commits to the blinded variant of $r(X)$, *i.e.,* $\hat{r}(X) = r(X) + m(X)z_H(X)$. Then we develop a zero-knowledge argument of knowledge for the following relation $\mathcal{R}_{\mathsf{nz}}$ showing that $r(w^i) \neq 0$ for each $w^i \in \mathbb{H}$.

Round 1 Prover:

1. Sample random degree-1 blinding polynomial $b(X)$.
2. Compute $s(X)$ such that $s(w^i) = r(w^i)^{-1}, \forall w^i \in \mathbb{H}$.
3. Blind $s(X)$ by setting $\hat{s}(X) = s(X) + b(X)z_H(X)$.
4. Sample a random blinding polynomial $m(X)$ to blind $r(X)$ by setting $\hat{r}(X) = r(X) + m(X)z_H(X)$.
5. Compute $q(X)$ such that $\hat{r}(X)\hat{s}(X) - 1 = q(X)z_H(X)$.
6. Send $\mathtt{s} = [\hat{s}(\tau)]_1, \mathsf{q} = [q(\tau)]_1$.

Round 1 Verifier: Sample and send random $\gamma \in \mathbb{F}$.
Round 2 Prover: Send openings $r_\gamma = \hat{r}(\gamma), s_\gamma = \hat{s}(\gamma)$.
Round 2 Verifier: Sample and send random $\alpha \in \mathbb{F}$.
Round 3 Prover:

1. Set $\phi(X) = r(X) + \alpha\hat{s}(X) + \alpha^2 q(X)$.
2. Compute $(\cdot, \mathtt{t}) = \mathsf{KZG.Open}(\mathsf{SP}, \phi(X), \gamma)$.
3. Send $\mathtt{t}$.

Round 3 Verifier:

1. Compute $q_\gamma = (r_\gamma \cdot s_\gamma - 1) \cdot z_H(\gamma)^{-1}$.
2. Set $y = r_\gamma + \alpha s_\gamma + \alpha^2 q_\gamma$ and $\mathsf{c} = \mathtt{r} + \alpha\mathtt{s} + \alpha^2\mathsf{q}$.
3. Assert $1 = \mathsf{KZG.Verify}(\mathsf{SP}, \mathsf{c}, \gamma, y, \mathtt{t})$.

Figure 3: Interactive zero-knowledge argument of knowledge protocol Π_{nz} for relation $\mathcal{R}_{\mathsf{nz}}$.

$$\mathcal{R}_{\mathsf{nz}} = \left\{ \left(\begin{array}{c} (\mathsf{SP}, \mathbb{H}); \\ (\mathtt{r}); \\ (r(X), m(X)); \end{array} \right) \middle| \begin{array}{c} \mathtt{r} = [r(\tau) + m(\tau)z_H(\tau)]_1, \\ r(w^i) \neq 0 \ \forall w^i \in \mathbb{H} \end{array} \right\}$$

We provide the corresponding argument Π_{nz} in Figure 3.
Lemma 5. The protocol Π_{nz} for relation $\mathcal{R}_{\mathsf{nz}}$, see Figure 3, satisfies completeness, soundness, and zero-knowledge.

We provide the proof for Lemma 5 in Appendix B.3.

Proof π_{b_i} [Step 8]. The goal of this proof is to show that $\boldsymbol{b}_i$ is a valid unary encoding of bidder i's bid b_i. To construct this proof we use logarithmic derivatives, see Section 2.5.3, and translate the problem to an equality of sum of reciprocals. To do this, we first re-formulate the problem to make it more amenable to logarithmic derivatives, show its equivalency with the problem of unary encoding and then present the proof system for this re-formulated problem. Given a positive integer n, the unary representation for a value $b \in [0, n]$ includes an array of $|b|$ 'ones' and $|n - b|$ 'zeroes'. To prove that a vector $\boldsymbol{b}$ of length n is a valid unary encoding of b, consider the following lemma:

Lemma 6. There are $1 \leq i < n$ and $0 \leq j < n$ such that $i + j = n$ and $\boldsymbol{b}$ is a vector of i 'ones' concatenated with j 'zeros' if and only if $\boldsymbol{b}$ starts with '1' and vector Δ defined as below consists of all 'zeros' except a single 'one', *i.e.,* $\Delta_i = b_i - b_{i+1}$ if $i < n$ and $\Delta_i = b_i$ if $i = n$.

We provide the proof for Lemma 6 in Appendix B.4.

To prove the correct form of Δ, we use logarithmic derivatives [29]. Due to the unique fractional decomposition of logarithmic derivatives, see Section 2.5.3, we need to prove that $\sum_{i=1}^{n} \frac{1}{X+\Delta_i} = \frac{n-1}{X+0} + \frac{1}{X+1}$. Logarithmic derivatives allow us to check that, given the vectors $\boldsymbol{t}, \boldsymbol{f}$, and $\boldsymbol{m}$, each value f_i appears exactly m_i times in $\boldsymbol{t}$ which can be expressed via $\sum_{i=1}^{n} \frac{1}{X+f_i} = \sum_{i=1}^{n} \frac{m_i}{X+t_i}$. To prove this, we turn the fractional expression into a polynomial one via low degree extensions [1] of the functions $t_{|\mathbb{K}}, f_{|\mathbb{H}}, m_{|\mathbb{K}}$ over multiplicative subgroups $\mathbb{H}$ and $\mathbb{K}$ of size n. The prover can show that the equality holds at a random challenge γ as per Lemma 2, *i.e.,* that $\sum_{i=1}^{n} \frac{1}{\gamma+f(w^i)} = \sum_{i=1}^{n} \frac{m(w^i)}{\gamma+t(w^i)}$. With that in mind, the relation $\mathcal{R}_{\mathsf{unary}}$ for which we need to develop an argument of knowledge is as follows:

$$\mathcal{R}_{\mathsf{unary}} = \left\{ \left(\begin{array}{c} (\mathsf{SP}, \mathbb{H}); \\ (\mathtt{u}); \\ (u(X), m(X)); \end{array} \right) \middle| \begin{array}{c} \mathtt{u} = [u(\tau) + m(\tau) z_H(\tau)]_1 \\ u(X) \text{ is LDE of } \boldsymbol{b} \end{array} \right\}$$

We present the corresponding argument Π_{unary} in Figure 4.

Lemma 7. The protocol Π_{unary} for relation $\mathcal{R}_{\mathsf{unary}}$, see Figure 4, satisfies completeness, soundness, and zero-knowledge.

We provide the proof for Lemma 7 in Appendix B.5.

4.2 Bidding Phase

Proof π_{Z_i} [Step 2]. The goal of this proof is to enable a bidder to show that $\boldsymbol{Z} = (\boldsymbol{x} + \boldsymbol{b} \circ \boldsymbol{r}) \circ \boldsymbol{Y}$. More generally, given the message received after the preprocessing phase $\boldsymbol{Y}$, the bidder has to prove that their message sent in the bidding phase is of the form $\boldsymbol{Z} = \boldsymbol{a}\boldsymbol{Y}$, where $\boldsymbol{a} = \boldsymbol{x} + \boldsymbol{b} \circ \boldsymbol{r}$. We prove this by using an inner product argument (IPA) showing that $\langle \langle \boldsymbol{r} \circ \boldsymbol{a} \rangle, \boldsymbol{Y} \rangle = \langle \boldsymbol{r}, \boldsymbol{Z} \rangle$, where $\boldsymbol{r} = (r, r^2, \ldots, r^n)$ for a (separation) challenge r. [2] This way, the verifier computes $C = \sum r^j Z_j$ and the prover is left to show it is indeed equivalent to a generalized IPA [13] between $\boldsymbol{a}$ and $\boldsymbol{Y}$. Thus, we need to develop an argument of knowledge for the following relation:

$$\mathcal{R}_{\mathsf{sum}} = \left\{ \left(\begin{array}{c} (\mathsf{SP}, G, H, \mathbb{H}); \\ (\mathtt{a}, \boldsymbol{Y}, \boldsymbol{Z}); \\ (\boldsymbol{a}) \end{array} \right) \middle| \begin{array}{c} a(X) \text{ is LDE of } \boldsymbol{a}, \\ \mathtt{a} = [a(\tau)]_1, \\ a(w^j)\boldsymbol{Y}_j = \boldsymbol{Z}_j \end{array} \right\}$$

We provide the corresponding argument Π_{sum} in Figure 5.

Lemma 8. The protocol Π_{sum} for relation $\mathcal{R}_{\mathsf{sum}}$, see Figure 5, satisfies completeness, soundness, and zero-knowledge.

[1]Let $\phi_1 : \{0,1\}^v \to \mathbb{F}$ be any function mapping a v-dimensional Boolean hypercube to $\mathbb{F}$. A v-variate polynomial ϕ_2 over $\mathbb{F}$ is said to be an extension of ϕ_1 if ϕ_2 agrees with ϕ_1 for all $\boldsymbol{x} \in \{0,1\}^v$. We can think of a (low-degree) extension ϕ_2 of ϕ_1 as an error-corrected encoding, amplifying the distance between the original polynomials according to Schwartz-Zippel lemma. We refer the reader to [47] for more details.

[2]A random linear combination of terms is needed to guarantee that the equivalency holds for all $j \in [n]$ and no terms cancel out.

Round 1 Prover:

1. Compute $f(X)$ such that $f(w^i) = u(w^i) - u(w^{i+1}), \forall w^i \in \mathbb{H} \setminus w^{n-1}$.
2. Sample blinding polynomials $m(X), b(x)$ to blind $u(X)$ by setting $\hat{u}(X) = u(X) + m(X) z_H(X)$ and $f(X)$ by setting $\hat{f}(X) = f(X) + b(X) z_H(X)$.
3. Compute $(\cdot, \mathtt{u}_1) = \mathsf{KZG.Open}(\mathsf{SP}, \hat{u}(X), 1)$.
4. Compute $Q_f(X)$ such that $\hat{f}(X) - (u(X) - u(wX)) = Q_f(X) z_H(X)$.
5. Send $\mathtt{f} = [\hat{f}(\tau)]_1, \mathtt{u}_1, \mathtt{q_f} = [Q_f(\tau)]_1$.

Round 1 Verifier: Sample and send random γ.

Round 2 Prover:

1. Compute $B(X)$ such that $B(w^i) = \frac{1}{\gamma + f(w^i)}, \forall w^i \in \mathbb{H}$.
2. Sample random $r(X)$ and compute blinded $\hat{B}(X) = B(X) + r(X) z_H(X)$.
3. Compute Q_B that $\hat{B}(X)(f(X) + \gamma) - 1 = Q_B(X) z_H(X)$.
4. Sample random $S(X)$ and set $s = \sum_{h \in \mathbb{H}} S(h)$.
5. Send $\mathtt{b} = [\hat{B}(\tau)]_1, \mathtt{q_b} = [Q_B(\tau)]_1, \mathtt{s} = [S(\tau)]_1, s$.

Round 2 Verifier: Sample and send random α.

Round 3 Prover:

1. Set $v = \frac{1}{\gamma} + \frac{n-1}{\gamma+1}$.
2. Compute $R(X)$ and $Q(X)$ such that $S(X) + \alpha \hat{B}(X) = \frac{s + \alpha \cdot v}{n} + X R(X) + Q(X) z_H(X)$.
3. Send $\mathtt{r} = [R(\tau)]_1, \mathtt{q} = [Q(\tau)]_1$.

Round 3 Verifier: Sample and send random β.

Round 4 Prover: Send openings $u_\beta = u(\beta), u_{w,\beta} = u(w\beta)$, $f_\beta = \hat{f}(\beta)$, $Q_{f,\beta} = Q_f(\beta)$, $B_\beta = \hat{B}(\beta)$, $s_\beta = S(\beta)$, $Q_{B,\beta} = Q_B(\beta)$, $R_\beta = R(\beta)$, and $Q_\beta = Q(\beta)$.

Round 4 Verifier: Sample and send random v.

Round 5 Prover:

1. Compute $A(X) = (u(X) + v f(X) + v^2 Q_f(X) + v^3 \hat{B}(X) + v^4 S(X) + v^5 Q_B(X) + v^6 R(X) + v^7 Q(X)) \cdot (X + \beta)^{-1}$.
2. Compute $A_w(X) = u(X) \cdot (X - w\beta)^{-1}$.
3. Send $\mathtt{a} = [A(\tau)]_1$ and $\mathtt{a_w} = [A_w(\tau)]_1$.

Round 5 Verifier:

1. Assert that $1 = \mathsf{KZG.Verify}(\mathsf{SP}, \mathtt{u}, 1, 1, \mathtt{u}_1)$ and $f_\beta - (u_\beta - u_{w,\beta}) = Q_{f,\beta} z_H(\beta)$
2. Set $v = \frac{1}{\gamma} + \frac{n-1}{\gamma+1}$.
3. Assert that $S(\beta) + \alpha B_\beta = \frac{s + \alpha \cdot v}{n} + \beta R_\beta + Q_\beta z_H(\beta)$.
4. Set $C = \mathtt{u} + v\mathtt{f} + v^2 \mathtt{q_f} + v^3 \mathtt{b} + v^4 \mathtt{s} + v^5 \mathtt{q_b} + v^6 \mathtt{r} + v^7 \mathtt{q}$ and $y = u_\beta + v f_\beta + v^2 Q_{f,\beta} + v^3 B_\beta + v^4 s_\beta + v^5 Q_{B,\beta} + v^6 R_\beta + v^7 Q_\beta$.
5. Assert that $1 = \mathsf{KZG.Verify}(\mathsf{SP}, C, \beta, y, \mathtt{a})$ and $1 = \mathsf{KZG.Verify}(\mathsf{SP}, \mathtt{u}, w\beta, u_{w,\beta}, \mathtt{a_w})$.

Figure 4: Interactive zero-knowledge argument of knowledge protocol Π_{unary} for relation $\mathcal{R}_{\mathsf{unary}}$.

```
**Verifier**: Send a random $c$.
**Prover and Verifier**:
  1. Compute $\mathbf{c} = (1, c, c^2, \dots, c^{n-1})$ and $C = \langle \mathbf{c}, \mathbf{Y} \rangle$.
  2. Set $C' = C$ and $n' = n$.
  3. Rescale the basis $\mathbf{X}$ by computing $\mathbf{X}_c = \mathbf{c} \circ \mathbf{X}$ .
**Prover**: Set $\mathbf{a}' = \mathbf{a}$ and $\mathbf{X}' = \mathbf{X}_c$ and initialize $r = 0$.
Until $n' = 1$ do:
  1. Set $n' = \frac{n'}{2}$, $L = \langle \mathbf{a}'_{lo}, \mathbf{X}'_{hi} \rangle$, and $R = \langle \mathbf{a}'_{hi}, \mathbf{X}'_{lo} \rangle$.
  2. Sample random $r_l$ and $r_r$.
  3. Send $L + r_l H$ and $R + r_r H$.
**Verifier**:
  4. Sample and send a random challenge $\alpha$.
  5. Compute $C' = \alpha^{-1} L + C' + \alpha R$.
**Prover**:
  6. Compute $C' = \alpha^{-1} L + C' + \alpha R$.
  7. Set $a' = \langle \mathbf{a}'_{lo}, \mathbf{a}'^{\alpha}_{hi} \rangle$ and $X' = \langle \mathbf{X}'_{lo}, \mathbf{X}'^{\alpha^{-1}}_{hi} \rangle$.
  8. Compute $r = r + \alpha^1 r_l + \alpha r_r$.
When $n' = 1$:
**Prover and Verifier**:
  1. Compute $f(X) = \prod_{i=0}^{l=\log n} (1 + \alpha_{l-i} X^{2^i})$.
  2. Interpolate $\tilde{f}(X)$ from $f(X)$.
**Verifier**:
  1. Compute $X' = \langle \mathbf{X}_c, \mathbf{f} \rangle$, where vector $\mathbf{f}$ denotes coefficients of $f(X)$.
  2. Invoke $\mathcal{R}_{\mathsf{pse}}((\mathbf{pp}, \mathbf{srs}, X', H); (q_a, \tilde{f}(X), C'); (a(X), a', r))$.
```

Figure 5: Interactive zero-knowledge argument of knowledge protocol Π_{sum} for relation $\mathcal{R}_{\mathsf{sum}}$.

We provide the proof for Lemma 8 in Appendix B.6.

Removal of one inner product. As part of the argument system for the above relation, we introduce a technique that could be of independent interest. We observe that when running an IPA between vectors of $\boldsymbol{a}$ and $\boldsymbol{Y}$ with the verifier having access to the polynomial commitment q_a of $\boldsymbol{a}$, one can replace an instance of IPA with an instance of univariate sumcheck [6]. As we describe next, it reduces the computation cost of the prover and the proof size. We first build an intuition on why this works followed by our full argument protocol for $\mathcal{R}_{\mathsf{sum}}$.

Assume that a prover $\mathcal{P}$ wants to convince a verifier $\mathcal{V}$ that $C = \langle \boldsymbol{a}, \boldsymbol{X} \rangle$, where $\boldsymbol{a} \in \mathbb{F}^n$ and $\boldsymbol{X} \in \mathbb{G}_1^n$ is a set of random generators. Let $F = \langle \boldsymbol{a}, \boldsymbol{G} \rangle$ be a commitment to the vector $\boldsymbol{a}$ using some basis $\boldsymbol{G} \in \mathbb{G}_1^n$. Then, $\mathcal{P}$ and $\mathcal{V}$ should run two IPAs in parallel[3], one for F and the other for C. The former is needed to convince $\mathcal{V}$ that $\mathcal{P}$ is indeed using $\boldsymbol{a}$. Following the IPA protocol [12, 13], at each round the prover folds in half the vectors $\boldsymbol{a}, \boldsymbol{G}$ and $\boldsymbol{X}$ and both $\mathcal{P}$ and $\mathcal{V}$ derive updated commitments F', C'. In the last round, $\mathcal{P}$ sends a fully folded a', and $\mathcal{V}$ computes the fully folded values G', X'. The verifier then checks if $a'G' = F'$ and $a'X' = C'$. It turns out that computing a' from $\boldsymbol{a}$ is a multi-scalar multiplication of size n where the scalar factors are the coefficients of the polynomial $f(X) = \prod_{i=0}^{l=\log n} (1 + \alpha_{l-i} X^{2^i})$, and α_i are the random challenges picked by the verifier [11, 13][4]. So, we have $a' = \langle \boldsymbol{a}, \boldsymbol{f} \rangle$ where $\boldsymbol{f}$ are the coefficients of polynomial $f(X)$. Further, let $\tilde{f}(X)$ denote a polynomial such that its evaluations are equal to $\boldsymbol{f}$. Therefore, instead of running the IPA for F the prover needs to show that $\sum_{i=0}^{n} a(X)\tilde{f}(X) = a'$ using an instance of univariate sumcheck. This way, the prover no longer needs to compute the folding for G which is genuinely expensive, overall improving run time. Note that we instantiate the polynomial commitment scheme with KZG to show that $\sum_{h \in \mathbb{H}} a(h)\tilde{f}(h) = a'$, where $\mathbb{H}$ is a multiplicative subgroup of $\mathbb{F}$ of order n.

So far, we have not considered zero-knowledge. To make IPA zero-knowledge, the prover first forms the perfectly blinded commitment $C_b = C + rH$ and then samples random scalars r_l, r_r to compute the left and right blinded commitments $L_{i,b} = L_i + r_l H$, $R_{i,b} = R_i + r_r H$. One subtlety we need to get around is to avoid sending a' in plain at the last round while allowing the verifier to check that $a'X' = C'$ holds. We tackle this by having the prover show knowledge of opening of a Pedersen commitment that is defined by the folded basis X' and blinder H. Thus, we should develop an argument $\mathcal{R}_{\mathsf{pse}}$ to link an instance of the Pedersen commitment with a univariate sumcheck showing that given a Pedersen commitment $P = xG + rH$, we have $\sum_{i=0}^{n} a(X)\tilde{f}(X) = x$ without leaking any information about x.

Remark 1. The auditor is not computationally restricted thus we assume they can interpolate $\tilde{f}(X)$ from $f(X)$. However, this part can be verifiably delegated. Let W denote a set of n-th roots of unity, then the prover can show that $\sum_{w^i \in W} \tilde{f}(w^i) p(w^i) = f(\alpha)$ for a random α such that $p(w^i) = \alpha^i$. Note that evaluating $f(X)$ at α takes $O(\log n)$ field operations, and validity of $p(X)$ can be proved by showing that $p(w^0) = 1$ and that $p(wX) = \alpha p(X)$ when $X \in W \setminus w^{n-1}$.

Remark 2. We develop $\mathcal{R}_{\mathsf{pse}}$ as a generic independent relation where both $a(X), b(X)$ are witnesses. However, in the last round of our inner product protocol we use the public polynomial $\tilde{f}(X)$, thus we slightly abuse notation and we put $\tilde{f}(X)$ directly in the instance instead of committing to it.

Linking Pedersen commitment to univariate sumcheck. We now develop a zero knowledge argument of knowledge for proving the equality of Pedersen commitment [42] and univariate sumcheck [6] that may be of independent interest. Given a Pedersen Commitment $P = xG + rH$, the

[3]Running IPAs in parallel matters security-wise, as we need to use the same set of challenges for both [23].

[4]In the non-interactive variant the challenges are computed by applying a hash function modeled in random oracle on the messaged received from the prover in each round of IPA for $C = \langle \boldsymbol{a}, \boldsymbol{X} \rangle$ which is running in parallel.

prover $\mathcal{P}$ aims to prove the knowledge of $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}^n$ such that $\sum_{i=0}^{n} \boldsymbol{a}_i \boldsymbol{b}_i = x$, without leaking any information about x. We define the relation $\mathcal{R}_{\mathsf{pse}}$ as follows:

$$\mathcal{R}_{\mathsf{pse}} = \left\{ \left(\begin{array}{c} (\mathsf{SP}, G, H, \mathbb{H}); \\ (\mathtt{a}, \mathtt{b}, P); \\ (a(X), b(X), x, r); \end{array} \right) \middle| \begin{array}{c} \mathtt{a} = [a(\tau)]_1 \\ \mathtt{b} = [b(\tau)]_1 \\ P = xG + rH \\ \sum_{h \in \mathbb{H}} a(h)b(h) = x \end{array} \right\}$$

We present a zero-knowledge argument realizing the relation $\mathcal{R}_{\mathsf{pse}}$ in Figure 6. At a high level, the prover $\mathcal{P}$ computes and commits to $a(X)$ and $b(X)$, low degree extensions of $\boldsymbol{a}$ and $\boldsymbol{b}$ over $\mathbb{H}$. Then the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ engage in a (public coin) interactive argument for proving knowledge of x and that $\sum_{h \in \mathbb{H}} a(h)b(h) = x$. This is done by invoking both (zero-knowledge) arguments for proving the opening of Pedersen commitment and univariate sumcheck. Note that proving the knowledge of opening of Pedersen Commitment is done by providing two Schnorr proofs of knowledge of discrete logarithm. However, $\mathcal{P}$ should take additional care when sampling blinders for the univariate sumcheck instance. The key insight is that in the first round $\mathcal{P}$ samples a blinder $B(X)$ for zero-knowledge univariate sumcheck such that its sum over $\mathbb{H}$ is equal to the randomness/blinder used in the first instance of the underlying Schnorr proof, denoted by p. Then in the second round after receiving challenge c from $\mathcal{V}$, $\mathcal{P}$ shows that $\sum_{h \in \mathbb{H}} B(h) + ca(h)b(h) = cx + p$. As mentioned earlier, we make use of this argument as a sub-argument of the protocol realizing the relation $\mathcal{R}_{\mathsf{pse}}$. So, we slightly modify the relation and assume that $\mathcal{P}$ already committed to $a(X), b(X)$. This, however, leads to the following issue. At the last round of the zero-knowledge univariate sumcheck, $\mathcal{P}$ has to send openings of $a(X), b(X)$ at a random challenge, potentially introducing a leakage affecting zero-knowledge. One way to deal with this is to assume that witness polynomials $a(X), b(X)$ are already properly blinded and thus openings do not produce any leakage. By looking closely at the argument for $\mathcal{R}_{\mathsf{pse}}$, we notice that in our case $a(x)$ is properly blinded and $b(x)$ is a public polynomial, guaranteeing zero-knowledge. So, we state the relation assuming the openings of $a(X), b(X)$ do not affect zero-knowledge. However, we later describe how to adapt the argument to make it generalized and cover the situations where the witness polynomials are committed without blinders. The argument is agnostic to the polynomial commitment scheme but we instantiate it with KZG.

Lemma 9. The protocol Π_{pse} for relation $\mathcal{R}_{\mathsf{pse}}$ presented in Figure 6 satisfies completeness, soundness, and zero-knowledge.

We provide the proof for Lemma 9 in Appendix B.8.

Proof π_w [Step 6]. After determining the highest bid w, the winner can simply open their bid vector at the corresponding position showing that $\boldsymbol{b}_w = 1$. We can further preserve the privacy of candidate winners, particularly in case there is a tie and only one of them should be picked. We now develop

Round 1 Prover:

1. Sample random p, s and compute $Q = pG + sH$.
2. Compute $b_0 = \frac{p}{n}$ and sample random $b_1, b_2, b_3, b_4, b_5, b_6$, then compute $B(X) = b_0 + b_1X + b_2X^2 + b_3X^3 + b_4X^4 + (b_5 + b_6X)z_H(X)$, observe that $\sum_{h \in \mathbb{H}} B(h) = p$.
3. Send $Q, \mathtt{B} = [B(\tau)]_1$.

Round 1 Verifier: Send random challenge c

Round 2 Prover:

1. Compute $z_1 = cx + p$ and $z_2 = cr + s$.
2. Compute $R(X), q(X)$ such that $B(X) + ca(X)b(X) = \frac{z_1}{|\mathbb{H}|} + XR(X) + q(X)z_H(X)$.
3. Compute $D(X) = R(X) \cdot X^{N-1-(n-2)}$ where N is the length of the SP proving key.
4. Send $z_1, z_2, \mathtt{R} = [R(\tau)]_1, \mathtt{q} = [q(\tau)]_1, \mathtt{D} = [D(\tau)]_1$.

Round 2 Verifier: Sample and send a random opening challenge $\gamma \in \mathbb{F}^* \setminus \{\mathbb{H}, 0\}$.

Round 3 Prover: Send openings $a_\gamma = a(\gamma), b_\gamma = b(\gamma), B_\gamma = B(\gamma), R_\gamma = R(\gamma)$.

Round 3 Verifier: Send random separation challenge v.

Round 4 Prover:

1. Compute $W(X) = (a(X) + vb(X) + v^2B(X) + v^3R(X) + v^4q(X)) \cdot (X - \gamma)^{-1}$.
2. Send $\mathtt{W} = [W(\tau)]_1$.

Round 4 Verifier:

1. Check that $cP + Q = z_1G + z_2H$.
2. Compute $q_\gamma = (B_\gamma + ca_\gamma b_\gamma - \frac{z_1}{n} - \gamma R_\gamma) \cdot z_H(\gamma)^{-1}$.
3. Compute $C = \mathtt{a} + v\mathtt{b} + v^2\mathtt{B} + v^3\mathtt{R} + v^4\mathtt{q}$.
4. Compute $y = a_\gamma + vb_\gamma + v^2B_\gamma + v^3R_\gamma + v^4q_\gamma$.
5. Assert $1 = \mathsf{KZG.Verify}(\mathsf{SP}, C, \gamma, y, \mathtt{W})$.
6. Set degree bound $d = N - 1 - (n-2)$, where N is the length of the SP proving key.
7. Assert $e(\mathtt{R}, [x^d]_2) = e(\mathtt{D}, [1]_2)$.

Figure 6: Interactive zero-knowledge argument of knowledge protocol Π_{pse} for relation $\mathcal{R}_{\mathsf{pse}}$.

another zero-knowledge argument to enable the candidate winner to show their eligibility while preserving their privacy. Before describing the relation, recall that the followings hold: (1) $\boldsymbol{X} = \boldsymbol{x}G$, (2) $\boldsymbol{b}$ is a vector with valid unary encoding, (3) $\boldsymbol{a} = \boldsymbol{x} + \boldsymbol{b} \circ \boldsymbol{r}$, and (4) $\boldsymbol{Z} = \boldsymbol{aY}$. We observe that the candidate bidder can demonstrate their eligibility if they manage to prove that the discrete logs of $\boldsymbol{X}_w$ and $\boldsymbol{Z}_w$ are different, ensuring that $\boldsymbol{b}_w = 1$. So, it is enough to prove knowledge of x, and r such that $\boldsymbol{X}_w = xG$, $(x + r)\boldsymbol{Y}_w = \boldsymbol{Z}_w$, and $r \neq 0$. To preserve privacy, we follow the standard approach in the literature [45] to have auctioneer/auditor compute a public vector commitment C (*e.g.,* using a Merkle Tree) of all triples $(\boldsymbol{X}_w, \boldsymbol{Z}_w, \boldsymbol{Y}_w)$ and then each candidate winner proves knowledge of satisfying x and r for some pair in the vector. Further, each candidate winner has to compute a nullifier [45] nul to ensure that they cannot submit multiple proofs. The corresponding relation $\mathcal{R}_{\mathsf{win}}$ is as follows:

$$\mathcal{R}_{\mathsf{win}} = \left\{ \left(\begin{array}{c} C, nul; \\ x, r, X, Y, Z, \pi \end{array} \right) \middle| \begin{array}{c} r \neq 0, \\ xG = X, \\ (x+r)Y = Z, \\ nul = \mathcal{H}(x, r), \\ \mathsf{Verify}(C; (X, Y, Z); \pi) = 1 \end{array} \right\}$$

Observe that it is enough to prove knowledge of some x, and r and not the ones committed in the first place. This follows from the fact that the bidder already proved connection between vectors $\boldsymbol{x}$ and $\boldsymbol{r}$ with $\boldsymbol{X}$ and $\boldsymbol{Y}$. Should the bidder provide any other satisfying x' and r', it would break the discrete logarithm assumption. If there is a tie we can utilize public randomness [34] to break the tie and choose among the submitted proofs fairly. We highlight that preserving the privacy of the winner is important in applications where the winner could be subject to attacks like briberies [4].

5 Extensions

5.1 Efficient Second-price Auctions

Our main protocol, see Figure 1, can support both first- and second-price auctions. However, in the latter case, we need to re-run the protocol without the winner to detect the second highest price, as with other private auctions [57]. We now show how to modify the Cryptobazaar protocol to support second-price and generally $(p+1)$st-price auctions more efficiently without having to re-run the protocol. In this variant, the set of bidders who propose the top p bids will win and purchase the (identical) goods at the $(p+1)$st price.

Preprocessing. This phase is mostly as before, with the only exception being that each bidder uses a Boolean encoding for their bid b such that $\boldsymbol{b} = (0, \ldots, b_j, \ldots, 0)$, where $b_j = 1$ if and only if $j = b$. To construct validity proof π_{b_i}, we can use similar techniques based on log derivatives as before.

Finding the highest bid. This step is as before, and the highest bid is defined by the highest position w such that $\boldsymbol{R}_w \neq 0$.

Finding the winner. This step is as before, except that we now have a possible set of winners bidding at positions $[w, w-p+1]$, with the sale price being the highest position w' such that $\boldsymbol{R}_{w'} \neq 0$ and $w' < w-p+1$.

We remark that the efficiency gained in this variant comes with some privacy leakage. That is, one can learn all the bids submitted in the auction protocol by examining the protocol transcript, but without linking the bids to the corresponding bidders. This variant still offers a decent amount of privacy (*i.e.,* unlinkability of bidders and their bids [18]) and could be of interest depending on the applications, *e.g.,* when moving from single-item to multi-item NFT auctions [38].

5.2 Sequential First-price Auctions

An inherent limitation of the AV protocol is that the winner can examine the protocol transcript and check if they were the only winner. In particular, they can try inputting x in the second round and see if the output is still random, implying another party has vetoed. Consequently, the winner of a Cryptobazaar auction can learn the second-highest price of a given run. A similar issue has been also observed in other protocols like Addax [57]. Although this might not be problematic in a Vickery auction or when the participating bidders frequently change, it might result in a strategic advantage to the current winner when choosing their bids for future runs of iterative/sequential first-price auctions, *e.g.,* when several items are sold one after the other to the same group of buyers [22]. For example, the winner who learns the second-highest price might choose their bid in the next run slightly above the previous second-highest price to minimize the amount they have to pay while maximizing their winning chances.

Before we describe how to address this issue below, observe that there is a conflict of interest between the seller and the winner. Thus our solution aims to take advantage of such collusion disincentivization between the seller and a bidder preventing them from winning at a lower price. So, we assume the auctioneer has common interest with the seller (*e.g.,* auctioneer is the seller).

Preprocessing. This step is as before, except that the auctioneer also acts as a bidder and submits their own bid. In particular, for their own bid they choose the maximum price, *i.e.,* $b_0 = n$. Moreover, there is no need for the bidders to decide and commit to their bid in this phase, allowing them to *adaptively* decide on their bids in the bidding phase. This further allows running the pre-processing in an *offline* phase as common in the MPC literature [20].

Finding the highest bid. This step is as before. However, since the auctioneer bid the maximum price $b_0 = n$ the leakage of the second-highest price to the winner is prevented. The auctioneer then computes $\boldsymbol{Y}_0$ with $b_0 = 0$ locally to find the highest position w such that $\boldsymbol{R}_w \neq 0$ and announces this value.

Finding the winner. This step is as before, except that we might need to add a *fraud* proof. That is, if the auctioneer announces a bid w' that is lower than the actual highest bid w, the honest winner could present a fraud proof showing that $\boldsymbol{b}_w = 1$ to slash the cheating auctioneer.

Another option to relax the assumption on the auctioneer is to have multiple auctioneers and assume that at least one of them is honest in line with the anytrust threat model [49].

Re-using the preprocessing phase. To reduce the overheads for bidders in the iterative variant of Cryptobazaar, it is useful to explore whether bidders could re-use the values they computed in the preprocessing phase for future runs of the auction (besides the option of pre-computing k sets of these values in advance as soon as they know that they will participate in k

auctions). With that in mind, note that re-using the random matrix $\mathbf{Y}$ for multiple runs of the auction is not secure, as one can learn information about the bids by comparing the protocol transcripts. Interestingly, we can efficiently re-randomize the matrix $\mathbf{Y}$ by re-randomizing only a single row-vector $\boldsymbol{X}_i$ in the matrix $\mathbf{X}$. Given this, we can essentially make the protocol non-interactive. However, we need to be careful regarding the possible collusion between the party who re-randomizes the matrix $\mathbf{Y}$ and the bidders. We can either sample a small subset of bidders for re-randomization or make a threshold/anytrust security assumption on the auctioneer's side. In case some bidders wish to leave the auction, one could also on-board new bidders to the protocol and use their contributions for re-randomization without the need for bidders who remain to re-do their preprocessing. Moreover, observe that if we allow the winner to open its bid vector at the winning index w to announce its eligibility (instead of using a zero-knowledge set membership proof), the winner is incentivized to re-do its preprocessing phase to protect its privacy for future runs of the auction.

Protection against non-responsive winner. One might wonder what if a candidate winner does not show up to claim their win, given that there is no way to determine the winner if they do not come forward voluntarily due to the privacy guarantees of the AV protocol. We argue that this is not problematic since the winner has no incentive to remain silent. Furthermore, one can slightly tweak the protocol requiring the winner to claim their win within a certain time window and if they do not, declare the second-highest bidder as winner and give them the chance to claim the win. This continues until some bidder presents a valid proof for the currently eligible slot.

6 Evaluation

We implemented Cryptobazaar in Rust using the cryptography framework `arkworks` [59][5]. The implementation is generic and supports any pairing-friendly curve. We run our benchmarks on an Apple MacBook Pro with an M1 Max chip with 10 cores and 64 GB memory. For the benchmarks we instantiated our implementation with the BN254 curve and we focus on the main computational overheads for individual bidders and the auctioneer. We notice that the most demanding computation for both bidder and auctioneer is running multiscalar multiplications (MSM) which are implemented in the `arkworks` module `VariableBaseMSM`. The `arkworks` code is not particularly optimized and switching to hardware-specialized libraries such `RapidSnark` [43] or `gnark` [10] may further improve performance obviously.

Bidder overheads. The preprocessing phase of an individual bidder i is dominated by the computation of the validity proofs π_{x_i}, π_{r_i}, and π_{b_i} for relations $\mathcal{R}_{\mathsf{pv}}$, $\mathcal{R}_{\mathsf{nz}}$, and $\mathcal{R}_{\mathsf{unary}}$, respectively. All proofs require $O(1)$ elliptic curve points and field elements except the proof for $\mathcal{R}_{\mathsf{pv}}$. Thus, the amount of data each bidder has to send is $m+O(1)$ elliptic curve elements and $O(1)$ field elements. For example, given a price range of $n = 1024$ the overall amount of data an individual bidder has to send is about 32 KB assuming a single elliptic curve point is 32 bytes. The bidding phase of a bidder i is dominated by the computation of the validity proof π_{Z_i} for the relation $\mathcal{R}_{\mathsf{sum}}$. We provide the computational overheads to compute the above proofs for price ranges $n \in \{128, 1024, 8192\}$ in Table 1a.

Note that the computation of the validity proofs is not on the critical path for certain deployment scenarios given our threat model, like running an auction via optimistic roll-ups [44].

Auctioneer overheads. The preprocessing phase of the auctioneer is dominated by the computation of the vectors $\boldsymbol{Y}_i$ requiring 1 MSM and $m-1$ elliptic curve additions per AV and thus n MSMs of size m and $n \cdot (m-1)$ elliptic curve additions in total. We provide the running times to compute vectors $\boldsymbol{Y}_i$ for the number of bidders $m \in \{32, 128, 256\}$ and price ranges $n \in \{128, 1024, 8192\}$ in Table 1b. Each vector $\boldsymbol{Y}_i$ contains n elliptic curve points for $i \in [m]$ and assuming an elliptic curve point is 32 bytes, the auctioneer thus sends m vectors of size $32n$ bytes. For example, for $n = 1024$ and $m = 128$ this amounts to 32 KB per bidder or 4.2 MB in total.

After the bidding phase, the auctioneer needs to add n elliptic curve points per AV. It starts from the highest price and runs until it finds the first non-zero point. In the worst case where all bidders bid the minimal price, it runs $m \cdot n$ elliptic curve additions. We report the (worst case) running times for computing vector $\boldsymbol{R}$ for the number of bidders $m \in \{32, 128, 256\}$ and price ranges $n \in \{128, 1024, 8192\}$ in Table 1c. In practice the expected running time to compute $\boldsymbol{R}$ should be much lower since the auctioneer stops as soon as the first non-zero entry is found.

Remark. It would be helpful to see how the numbers we choose for evaluation reflect the real-world deployments. A promising application is the use of (sealed-bid) auctions in Ethereum via proposer-builder separation architecture [24]. In the current realization [22], builders take part in an auction to bid for their prepared block and the one with the highest bid is chosen to be proposed by the proposer. According to the recent work of [53], the block preparation is almost dominated by 25 builders/bidders ([53], Table 5). Moreover, the authors managed to collect 191 builder public keys ([53], Table 2), which means the number of bidders is likely lower given that each could hold several public keys (*e.g.,* Titan builder has disclosed 12 public keys in their official document). So, our experiments with 32, 128, and 256 bidders reflects what currently deployed large-scale systems like Ethereum require. Furthermore, bids range usually from cents to tens of dollars and a realistic deployment would likely be $1000 \leq n \leq 10000$ depending on the application [57].

[5]We provide our implementation at https://github.com/akinovak/cryptobazaar-impl.

Table 1: Cryptobazaar microbenchmarks (in ms) for number of bidders m and price ranges n.

(a) Individual bidder overheads to compute validity proofs.

n	128	1024	8192
π_{x_i}	13.59	101.51	807.21
π_{r_i}	2.38	10.25	58.38
π_{b_i}	2.53	10.68	62.03
π_{Z_i}	27.28	141.38	953.36

(b) Auctioneer overheads to compute AV matrix $\mathbf{Y}$.

m / n	128	1024	8192
32	1.84	14.19	112.12
128	4.29	38.87	286.60
256	7.75	59.00	552.24

(c) Auctioneer overheads to compute results vector $\boldsymbol{R}$.

m / n	128	1024	8192
32	0.30	6.36	50.48
128	1.95	26.83	145.06
256	4.01	32.90	265.14

Table 2: A high-level comparison among state-of-the-art sealed-bid auction protocols. A black (white) circle states that the protocol does (does not) provide a given property and a half circle states that the protocol provides the property under certain circumstances / with some restrictions. Privacy refers to the confidentiality of bids during and after protocol execution. Scalability refers to having low computation and communication costs. Trust minimization states whether the protocol makes any trust assumptions or not. Versatility captures the ability to (securely) support different protocol variants without major modifications.

Protocols	Privacy	Scalability	Trust minimization	Versatility
Riggs [48]	○	◐	●	○
Cicada [27]	○	◐	●	○
SEAL [3]	●	○	●	○
Addax [57]	●	●	○	◐
Cryptobazaar	●	●	●	●

7 Related Work

Riggs [48] realizes decentralized sealed-bid auctions using time-based cryptography. The protocol has bidders commit to their bids which are then either self-opened by bidders or force-opened through (sequential) computation. Although the idea of using time-based cryptography for sealed-bid auction was already explored in the literature [17, 52], this work addresses the practical details of the deployment setting including running auctions in parallel while locking up enough collateral without privacy leakage. Cicada [27] is another recent auction protocol that differs from Riggs by proposing a non-interactive protocol via homomorphic time lock puzzles [37] that pack many puzzles (*i.e.,* bids) into a single one. None of the aforementioned protocols offer privacy for bids after the protocol execution though. In Cicada, there is an on-chain coordinator (as auctioneer) and off-chain solver. The solver needs to present the outcome to the auctioneer (with proofs) to let the auctioneer announce the winner. One can use Cryptobazaar in a similar fashion by having an on-chain auctioneer (*i.e.,* smart contract) and off-chain coordinator to help with the computation.

Two recent private auction protocols are SEAL [3], and Addax [57]. The SEAL protocol is auctioneer-free and the bidders themselves jointly compute the highest bid by interacting with each other, leading to a communication bottleneck. Similar to ours, they also make use of anonymous veto [30] as their underlying primitive but in a quite different way. First, they modify the original AV protocol where each bidder needs to commit to two random values for each run of the protocol. Second, bidders use a modified AV to compute the logical-OR of their input bits starting from the most significant bid where the bidders dynamically change their input to the AV according to some decision rule in case the output of last the AV equals 1, limiting the usability of the system. An issue of this approach is that bidders learn during protocol execution that they lost and thus need to be incentivized to finish the protocol run. Moreover, SEAL uses traditional Sigma protocols instead of more efficient succinct validity arguments. Addax [57] is a private online ad exchange that has a sealed-bid auction protocol at its core. The protocol adopts an affine aggregatable encoding (AFE) introduced in Prio [16] allowing an auction to be conducted over secret-shared bids. Cryptobazaar shares some common properties with Addax including using price range, unary encoding of bids, and performing bitwise OR on inputs. However, the crucial advantage of our design is its more relaxed threat model of only requiring a single untrusted auctioneer as coordinator while Addax needs at least two non-colluding auctioneers due to their adoption of Prio-based techniques. Although a side-by-side comparison between Cryptobazaar and Addax is difficult due to the different settings (*e.g.,* evaluations on Apple M1 vs AWS, threat models, etc.), we provide some numbers for intuition: Addax (Cryptobazaar) incurs a computational overhead of 1802 ms (1880 ms) per bidder per auction run and terminates in 440 ms (431 ms) for 96 (128) bidders and a price range of 10000 (8192). This shows that the performance of Cryptobazaar and Addax is in the same ballpark while Cryptobazaar provides a better threat model and more versatility in terms of deployment.

Acknowledgements

We thank Guillermo Angeris for constructive discussions on auction theory. This work is supported by a grant from the Ecosystem Support Program of the Ethereum Foundation.

References

[1] Ramiro Alvarez and Mehrdad Nojoumian. Comprehensive survey on privacy-preserving protocols for sealed-bid auctions. *Computers & Security*, 88:101502, 2020.

[2] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. *Journal of Cryptology*, 23(2):281–343, 2010.

[3] Samiran Bag, Feng Hao, Siamak F Shahandashti, and Indranil Ghosh Ray. Seal: Sealed-bid auction without auctioneers. *IEEE Transactions on Information Forensics and Security*, 15:2042–2052, 2019.

[4] Jeb Bearer, Benedikt Bünz, Philippe Camacho, Binyi Chen, Ellie Davidson, Ben Fisch, Brendon Fish, Gus Gutoski, Fernando Krell, Chengyu Lin, et al. The espresso sequencing network: Hotshot consensus, tiramisu data-availability, and builder-exchange. *Cryptology ePrint Archive*, 2024.

[5] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In *45th international colloquium on automata, languages, and programming (icalp 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[6] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P Ward. Aurora: Transparent succinct arguments for r1cs. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I 38*, pages 103–128. Springer, 2019.

[7] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *Theory of Cryptography: 14th International Conference, TCC 2016-B, Beijing, China, October 31-November 3, 2016, Proceedings, Part II 14*, pages 31–60. Springer, 2016.

[8] Erik-Oliver Blass and Florian Kerschbaum. Strain: A secure auction for blockchains. In *Computer Security: 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part I 23*, pages 87–110. Springer, 2018.

[9] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*, pages 327–357. Springer, 2016.

[10] Gautam Botrel, Thomas Piellard, Youssef El Housni, Arya Tabaie, Gus Gutoski, and Ivo Kubjas. Consensys/gnark-crypto: v0.11.2, January 2023.

[11] Sean Bowe, Jack Grigg, and Daira Hopwood. Recursive proof composition without a trusted setup. *Cryptology ePrint Archive*, 2019.

[12] 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 (SP)*, pages 315–334. IEEE, 2018.

[13] Benedikt Bünz, Mary Maller, Pratyush Mishra, Nirvan Tyagi, and Psi Vesely. Proofs for inner pairing products and applications. In *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part III 27*, pages 65–97. Springer, 2021.

[14] Dario Catalano and Dario Fiore. Vector commitments and their applications. In *Public-Key Cryptography–PKC 2013: 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26–March 1, 2013. Proceedings 16*, pages 55–72. Springer, 2013.

[15] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. Marlin: Preprocessing zksnarks with universal and updatable srs. In *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I 39*, pages 738–768. Springer, 2020.

[16] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *14th USENIX symposium on networked systems design and implementation (NSDI 17)*, pages 259–282, 2017.

[17] Dominic Deuber, Nico Döttling, Bernardo Magri, Giulio Malavolta, and Sri Aravinda Krishnan Thyagarajan. Minting mechanism for proof of stake blockchains. In

Applied Cryptography and Network Security: 18th International Conference, ACNS 2020, Rome, Italy, October 19–22, 2020, Proceedings, Part I 18, pages 315–334. Springer, 2020.

[18] Jannik Dreier, Pascal Lafourcade, and Yassine Lakhnech. Formal verification of e-auction protocols. In *International Conference on Principles of Security and Trust*, pages 247–266. Springer, 2013.

[19] Youssef El Housni and Gautam Botrel. Edmsm: multi-scalar-multiplication for snarks and faster montgomery multiplication. *Cryptology ePrint Archive*, 2022.

[20] Daniel Escudero, Vipul Goyal, Antigoni Polychroniadou, and Yifan Song. Turbopack: honest majority mpc with constant online communication. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 951–964, 2022.

[21] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer, 1986.

[22] Flashbots. Introduction to mev-boost. https://docs.flashbots.net/flashbots-mev-boost/introduction, 2024.

[23] Ethereum Foundation. Curdleproofs: A shuffle argument protocol. https://github.com/asn-d6/curdleproofs/tree/main, 2022.

[24] Ethereum Foundation. Proposer builder separation (pbs) - ethereum roadmap. https://ethereum.org/en/roadmap/pbs/, 2024.

[25] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part II 38*, pages 33–62. Springer, 2018.

[26] Ariel Gabizon, Zachary J Williamson, and Oana Ciobotaru. Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *Cryptology ePrint Archive*, 2019.

[27] Noemi Glaeser, István András Seres, Michael Zhu, and Joseph Bonneau. Cicada: A framework for private non-interactive on-chain auctions and voting. *Cryptology ePrint Archive*, 2023.

[28] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[29] Ulrich Haböck. Multivariate lookups based on logarithmic derivatives. *Cryptology ePrint Archive*, 2022.

[30] Feng Hao and Piotr Zieliński. A 2-round anonymous veto protocol. In *International Workshop on Security Protocols*, pages 202–211. Springer, 2006.

[31] J Horwitz and K Hagey. Google's secret 'project bernanke'revealed in texas antitrust case. *Wall Street Journal*, 2021.

[32] Jong-Hyuk Im, Taek-Young Youn, and Mun-Kyu Lee. Privacy-preserving blind auction protocol using fully homomorphic encryption. *Advanced Science Letters*, 22, 2016.

[33] Aniket Kate, Gregory M Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16*, pages 177–194. Springer, 2010.

[34] Alireza Kavousi, Zhipeng Wang, and Philipp Jovanovic. Sok: Public randomness. In *2024 IEEE 9th European Symposium on Security and Privacy (EuroS&P)*, pages 216–234. IEEE, 2024.

[35] Hiroaki Kikuchi. (m+ 1) st-price auction protocol. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, 85(3):676–683, 2002.

[36] Michal Król, Alberto Sonnino, Argyrios Tasiopoulos, Ioannis Psaras, and Etienne Rivière. Pastrami: privacy-preserving, auditable, scalable & trustworthy auctions for multiple items. In *Proceedings of the 21st International Middleware Conference*, pages 296–310, 2020.

[37] Giulio Malavolta and Sri Aravinda Krishnan Thyagarajan. Homomorphic time-lock puzzles and applications. In *Annual International Cryptology Conference*, pages 620–649. Springer, 2019.

[38] Jason Milionis, Dean Hirsch, Andy Arditi, and Pranav Garimidi. A framework for single-item nft auction mechanism design. In *Proceedings of the 2022 ACM CCS Workshop on Decentralized Finance and Security*, pages 31–38, 2022.

[39] Jose A Montenegro, Michael J Fischer, Javier Lopez, and Rene Peralta. Secure sealed-bid online auctions using discreet cryptographic proofs. *Mathematical and Computer Modelling*, 57(11-12):2583–2595, 2013.

[40] Valeria Nikolaenko, Sam Ragsdale, Joseph Bonneau, and Dan Boneh. Powers-of-tau to the people: Decentralizing setup ceremonies. In *International Conference on Applied Cryptography and Network Security*, pages 105–134. Springer, 2024.

[41] Anca Nitulescu. zk-snarks: A gentle introduction. *Ecole Normale Superieure*, 2020.

[42] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international cryptology conference*, pages 129–140. Springer, 1991.

[43] RapidSnark. https://github.com/iden3/rapidsnark, 2021.

[44] Ethereum Research. Nft auction. https://ethresear.ch/t/off-chain-l2-nft-auction-protocol-idea/12930, 2024.

[45] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE symposium on security and privacy*, pages 459–474. IEEE, 2014.

[46] Jan Christoph Schlegel. Transaction ordering auctions. *arXiv preprint arXiv:2312.02055*, 2023.

[47] Justin Thaler et al. Proofs, arguments, and zero-knowledge. *Foundations and Trends® in Privacy and Security*, 4(2–4):117–660, 2022.

[48] Nirvan Tyagi, Arasu Arun, Cody Freitag, Riad Wahby, Joseph Bonneau, and David Mazières. Riggs: Decentralized sealed-bid auctions. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 1227–1241, 2023.

[49] David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. Scalable anonymous group communication in the anytrust model. In *European Workshop on System Security (EuroSec)*, volume 4, 2012.

[50] Fei Wu, Thomas Thiery, Stefanos Leonardos, and Carmine Ventre. Strategic bidding wars in on-chain auctions. *arXiv preprint arXiv:2312.14510*, 2023.

[51] Pengcheng Xia, Haoyu Wang, Zhou Yu, Xinyu Liu, Xiapu Luo, and Guoai Xu. Ethereum name service: the good, the bad, and the ugly. *arXiv preprint arXiv:2104.05185*, 2021.

[52] Jie Xiong and Qi Wang. Anonymous auction protocol based on time-released encryption atop consortium blockchain. *arXiv preprint arXiv:1903.03285*, 2019.

[53] Sen Yang, Kartik Nayak, and Fan Zhang. Decentralization of ethereum's builder market. *arXiv preprint arXiv:2405.01329*, 2024.

[54] Shuai Yuan, Jun Wang, Bowei Chen, Peter Mason, and Sam Seljan. An empirical study of reserve price optimisation in real-time bidding. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1897–1906, 2014.

[55] Arantxa Zapico, Vitalik Buterin, Dmitry Khovratovich, Mary Maller, Anca Nitulescu, and Mark Simkin. Caulk: Lookup arguments in sublinear time. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 3121–3134, 2022.

[56] Haoqian Zhang, Michelle Yeo, Vero Estrada-Galinanes, and Bryan Ford. Zeroauction: Zero-deposit sealed-bid auction via delayed execution. *Cryptology ePrint Archive*, 2024.

[57] Ke Zhong, Yiping Ma, Yifeng Mao, and Sebastian Angel. Addax: A fast, private, and accountable ad exchange infrastructure. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI'23)*, pages 825–848, 2023.

[58] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *International symposium on symbolic and algebraic manipulation*, pages 216–226. Springer, 1979.

[59] zkSNARK ecosystem. arkworks contributors. https://arkworks.rs, 2024.

A Definitions

A.1 Zero Knowledge Argument of Knowledge

A proof system enables a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ about some statement u such that $\exists w : (u;w) \in \mathcal{R}$, where w is the corresponding witness and $\mathcal{R}$ is a polynomial-time decidable relation. A proof of knowledge system further convinces the verifier that not only the witness exists, but also the prover knows it. When a proof system only demonstrates some statement holds and does not leak any information about the witness, it is zero-knowledge. A proof system is an argument when it holds for a computationally bounded prover under certain computationally hard assumption.

Definition 3 (zkSNARK)**.** A (non-interactive) argument system $\mathsf{AS} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Simulate})$ for $\mathcal{R}$ is a zkSNARK if it satisfies the following properties:

- **Completeness.** Given a true statement u for relation $\mathcal{R}$, a honest prover $\mathcal{P}$ with a valid witness w should convince the verifier $\mathcal{V}$. More formally, for all $\lambda \in \mathbb{N}$ and for all $(u,w) \in \mathcal{R}$:

$$\Pr\left[\mathsf{AS.Verify}(\mathsf{crs}, u, \pi) = 1 \;\middle|\; \begin{array}{r} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{AS.Setup}(1^\lambda, \mathcal{R}) \\ \pi \leftarrow \mathsf{AS.Prove}(\mathsf{crs}, u, w) \end{array}\right] = 1$$

- **Knowledge Soundness.** There is an extractor that can compute a witness whenever the adversary produces a valid argument. The extractor gets full access to the adversary's state, including any random coins. Formally, we require that for all PPT adversaries $\mathcal{A}$ there exists a PPT extractor $\mathcal{E}_{\mathcal{A}}$ such that

$$\Pr\left[\begin{array}{r} \mathsf{AS.Verify}(\mathsf{crs},u,\pi)=1 \\ \wedge (u,w)\notin\mathcal{R} \end{array} \middle| \begin{array}{r} (\mathsf{crs},\mathsf{td})\leftarrow\mathsf{AS.Setup}(1^\lambda,\mathcal{R}) \\ ((u,\pi);w)\leftarrow\mathcal{A}||\mathcal{E}_{\mathcal{A}}(\mathsf{crs}) \end{array}\right]\leq\mathsf{negl}$$

- **Succinctness.** A non-interactive argument where the verifier runs in polynomial time $\lambda+|u|$ and the proof and the common reference string sizes are polynomial λ, is a fully succinct SNARK.
- **Statistical Zero-knowledge.** An argument is zero-knowledge if it does not leak any information besides the truth of the statement. Formally, if for all $\lambda\in\mathbb{N}$, for all $(u,w)\in\mathcal{R}$ and for all PPT adversaries $\mathcal{A}$, the following two distributions are statistically close:

$$D_0=\{\pi_0\leftarrow\mathsf{AS.Prove}(\mathsf{crs},u,w):(\mathsf{crs},\mathsf{td})\leftarrow\mathsf{AS.Setup}(1^\lambda,\mathcal{R})\}$$
$$D_1=\{\pi_1\leftarrow\mathsf{AS.Simulate}(\mathsf{crs},\mathsf{td},u):(\mathsf{crs},\mathsf{td})\leftarrow\mathsf{AS.Setup}(1^\lambda,\mathcal{R})\}$$

An argument of knowledge is *public coin* if all the messages sent form the verifier have uniform distribution and are independent from the ones received from the prover. This then allows to make the protocol non-interactive using Fiat-Shamir [21].

B Proofs

B.1 Proof for Theorem 1

An auction protocol takes as input m submissions $b_1,\dots,b_m$ in a domain χ and outputs the highest bid $b_w=\max\{b_1,\dots,b_m\}$ as the winning one. Inspired by [27], we now provide a formal definition for a (verifiable and private) sealed-bid auction Π_{Auction}.

Definition 4 (Selead-bid auction)**.** A sealed bid auction $\Pi_{\mathsf{Auction}}=(\mathsf{Setup},\mathsf{Seal},\mathsf{Eval},\mathsf{Verify})$ is defined with the following algorithms:

- $(\mathsf{SP})\leftarrow\mathsf{Auction.Setup}(1^\lambda)$: The algorithm takes as input a security parameter λ, and outputs system parameters SP.
- $(c_i,\pi_i)\leftarrow\mathsf{Auction.Seal}(\mathsf{SP},i,b)$: The algorithm takes as input the system parameters SP and submission b of user $i\in[m]$, and outputs a sealed bid c_i and a proof of well-formedness π_i.
- $(y,\pi)\leftarrow\mathsf{Auction.Eval}(\mathsf{SP},\{c_i,\pi_i\}_{i\in[m]})$: The algorithm takes as input the system parameters SP, the sealed bid c_i together with their corresponding proof π_i for $i\in[m]$, and outputs the final result y and a validity proof π.
- $(1/0)\leftarrow\mathsf{Auction.Verify}(\mathsf{SP},y,\pi_w)$: The algorithm verifies if the output y is indeed the correct result of the auction b_w.

Definition 5 (Completeness)**.** A sealed-bid auction Π_{Auction} satisfies completeness if the algorithm Eval outputs the highest bid b_w assuming all parties follow the protocol and any setup phase is performed correctly. More formally, for all $\lambda\in\mathbb{N}$ and for all $b_1,\dots,b_m\in\chi$

$$\Pr\left[y=b_w \middle| \begin{array}{r} (\mathsf{SP})\leftarrow\mathsf{Auction.Setup}(1^\lambda) \\ (c_i,\pi_i)\leftarrow\mathsf{Auction.Seal}(\mathsf{SP},i,b)\ \forall i\in[m] \\ (y,\pi)\leftarrow\mathsf{Auction.Eval}(\mathsf{SP},\{c_i,\pi_i\}_{i\in[m]}) \\ \mathsf{Auction.Verify}(\mathsf{SP},y,\pi_w)=1 \end{array}\right]=1$$

Definition 6 (Soundness)**.** Let b_w denote the highest (honest) bid and assume the adversary $\mathcal{A}$ corrupts the auctioneer and $k\leq m-2$ bidders. A sealed-bid auction Π_{Auction} satisfies soundness if there is a negligible function negl such that for all PPT adversaries $\mathcal{A}$ and for all $\lambda\in\mathbb{N}$

$$\Pr\left[\begin{array}{r} \mathsf{Auction.Verify}(\mathsf{SP},y,\pi_w)=1 \\ \wedge\, y\neq b_w \end{array} \middle| \begin{array}{r} (\mathsf{SP})\leftarrow\mathsf{Auction.Setup}(1^\lambda) \\ (b_j,c_j,\pi_j)\leftarrow\mathcal{A}(\mathsf{SP})\ \forall j\in[k] \\ (c,\pi)\leftarrow\mathsf{Auction.Seal}(\mathsf{SP},\cdot,b) \\ (y,\pi_w)\leftarrow\mathsf{Auction.Eval}(\mathsf{SP},\{c_i,\pi_i\}_{i\in[m]}) \end{array}\right]\leq\mathsf{negl}$$

Definition 7 (Privacy)**.** The sealed-bid auction Π_{Auction} satisfies privacy if for all PPT adversaries $\mathcal{A}$ corrupting the auctioneer and $k\leq m-2$ bidders and for all $\lambda\in\mathbb{N}$, there exists a PPT simulator $\mathcal{S}$ and a negligible function negl such that

$$\left|\Pr\left[\mathcal{A}(\mathsf{SP},b,c,\pi)=1 \middle| \begin{array}{l} (\mathsf{SP})\leftarrow\mathsf{Auction.Setup}(1^\lambda) \\ (c,\pi)\leftarrow\mathsf{Auction.Seal}(\mathsf{SP},\cdot,b)\ \forall i\notin[k] \end{array}\right]-\right.$$
$$\left.\Pr\left[\mathcal{A}(\mathsf{SP},b,c,\pi)=1 \middle| \begin{array}{l} (\mathsf{SP})\leftarrow\mathsf{Auction.Setup}(1^\lambda) \\ (b,c,\pi)\leftarrow\mathcal{S}(\mathsf{SP},\cdot)\ \forall i\notin[k] \end{array}\right]\right|\leq\mathsf{negl}$$

Proof. Completeness follows directly from that of the underlying AV protocol. Note that the value $Z=(x+b\circ r)\circ Y$ sent by each bidder in the bidding phase, see Figure 1, is essentially the same as that of sent in the original AV, where here the randomness r comes to play depending on the bid value. *Soundness* follows from the underlying argument of knowledge protocols. We now go over the possible situations that may lead to an incorrect outcome and argue that all are captured by the soundness of the arguments systems. The scenarios are: (1) the auctioneer is honest and some bidders are malicious, (2) the auctioneer is malicious and all bidders are honest, (3) the auctioneer is malicious and colludes with some bidders. A malicious bidder may (A1) provide inconsistent values for the first round of AV conflicting with their commitments, (A2) provide invalid unary encoding, (A3) provide inconsistent values for the second round of AV conflicting to their initial commitments, (A4) wrongly claim they are the candidate winner. All of the aforementioned items will lead to the failure of the verification of $\mathcal{R}_{\mathsf{pv}}$, $\mathcal{R}_{\mathsf{unary}}$, $\mathcal{R}_{\mathsf{sum}}$, and $\mathcal{R}_{\mathsf{win}}$ with overwhelming probability. A malicious auctioneer may (B1) send incorrect value after the first round of AV, (B2) output a wrong bid as the winner. Both of the aforementioned items will be detected publicly. A malicious auctioneer which is colluding with some malicious bidders could do a combination of the cases mentioned above that lead to failure in verification and are detected publicly.

Privacy: Before we describe our formal privacy analysis we intuitively argue about the privacy of the underlying AV protocol. When x_i is sampled randomly by bidder i, the adversary controlling all but one bidder (*i.e.,* without full collusion) cannot break the privacy of the bidder i (Theorem 4, [30]) and the confidentiality of its input. This is because the value Y_i has a uniform distribution (Lemma 3, [30]) and under the DDH assumption one cannot distinguish between x_iY_i and a random group element r_iY_i. It is straightforward to extend this to a vector of values $\boldsymbol{x}_i, \boldsymbol{r}_i$, and $\boldsymbol{Y}_i$ as in our case.

We now proceed to prove Cryptobazaar's privacy using real/ideal simulation paradigm [28]. We start by defining an ideal functionality $\mathcal{F}$ that acts as a trusted third party, receiving the protocol's input and giving its output. A protocol is said to be secure if what the adversary can learn from the interaction with protocol (and output) could also be learned from the interaction with the ideal functionality. This is formally shown by constructing a simulator $\mathcal{S}$ that can generate a simulated view (*i.e.,* protocol transcript) for the adversary which is indistinguishable from the actual protocol transcript without having access to the honest parties' inputs.

Let $b_1, \ldots, b_h$ denote the set of honest bids and without loss of generality assume the adversary $\mathcal{A}$ corrupts the auctioneer and $k \leq m-2$ bidders, with $h+k=m$. Further, let out = (b^*, sp) denote the set of outputs, including the highest (honest) bid, and second highest (honest) bid. The description of ideal functionality $\mathcal{F}$ and simulator $\mathcal{S}$ are given in Figure 7 and Figure 8, respectively. We define a sequence of hybrid distributions starting from the actual protocol transcript/distribution and ending with the simulated transcript/distributions. We argue that each two consecutive hybrids are computationally indistinguishable, implying the indistinguishably of the real and ideal distributions. Note that we exploit the zero-knowledge property of the underlying validity arguments, the privacy guarantees of the underlying AV protocol, and the hiding property of the underlying polynomial commitment to simulate the views.

Input: h bids $b_1, \ldots, b_h$ from honest bidders
Output: (b^*, sp), where

- Compute $b^* = \max(b_1, \ldots, b_h)$.
- Find w, such that $b_w = b^*$
- Compute $sp = \max(b_1, \ldots, b_{w-1}, b_{w+1}, \ldots, b_h)$.

$\mathcal{F}$ outputs (b^*, sp) at the end.

Figure 7: Ideal Functionality $\mathcal{F}$ for Cryptobazaar

Hybrid$_0$ This is the actual view of the Cryptobazaar protocol.

Hybrid$_1$ The same as Hybrid$_0$, except that the simulator $\mathcal{S}$ does the following on behalf of each honest bidder $i \in [h]$. It computes and appends $(\tilde{\mathrm{x}}, \tilde{\mathrm{q}}, \tilde{\mathrm{b}}, \tilde{\boldsymbol{X}}, \tilde{\pi}_x, \tilde{\pi}_r, \tilde{\pi}_b)$ to the public log. This view is indistinguishable from the last one due to the uniform random distribution of $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{r}}$, the zero-knowledge property of the underlying proofs for $\tilde{\pi}_x$, $\tilde{\pi}_r$, $\tilde{\pi}_b$, and the hiding property of the polynomial commitment $\tilde{\mathrm{b}}$. Note that the original KZG commitment is deterministic in the sense that the same polynomials have the same commitment. So, for the indistinguishably to hold we need to use blinded KZG.

Hybrid$_2$ The same as Hybrid$_1$, except that the matrix $\mathbf{Y}$ is computed with respect to the simulated row-vectors $\tilde{\boldsymbol{X}}_i$ for $i \in [h]$ and the ones from adversary $\boldsymbol{X}'_j$ for $j \in [k]$. This hybrid is computationally indistinguishable from Hybrid$_1$ due to the privacy guarantee of AV under the DDH assumption. That is, having one honest row-vector X included in the matrix $\mathbf{Y}$, its row-vectors $\boldsymbol{Y}_i$ have uniform distributions (Lemma 3, [30]).

Hybrid$_3$ The same as Hybrid$_2$, except that the simulator $\mathcal{S}$ computes the vectors $\tilde{\boldsymbol{Z}}_i$ on behalf of honest bidders using its chosen bids. It also computes the validity proofs $\tilde{\pi}_{Z_i}$ for $i \in [h]$. The indistinguishability of this hybrid from the previous one stems from the security guarantee of the AV protocol under the DDH assumption and the zero-knowledge property of the $\tilde{\pi}_{Z_i}$.

Hybrid$_4$ The same as Hybrid$_3$, except that the output vector is computed as $\boldsymbol{R} = \sum_{i=1}^{h}(\tilde{\boldsymbol{Z}}_i) + \sum_{j=1}^{k}(\boldsymbol{Z}'_j)$, where the highest position w should equal b^*. Otherwise, the simulator aborts. Observe that due to the unary encoding and the way AV works, a valid unary proof for the highest bid $\tilde{\pi}_{b^*}$ should be enough to ensure the privacy of honest bidders and failure to do so by $\mathcal{A}$ does not affect the security of the view. Therefore, this view is indistinguishable from the previous one.

Hybrid$_5$ The same as Hybrid$_4$, except that the simulator computes an eligibility proof $\tilde{\pi}_w$ on behalf of the candidate honest winner. This view is computationally indistinguishable form the previous one due to the zero-knowledge property of the $\tilde{\pi}_w$.

Hybrid$_6$ This is the view of $\mathcal{A}'$ simulated by $\mathcal{S}$ in Cryptobazaar. Further, the simulator sets $b^* = 0$ and sends the corresponding $\tilde{\boldsymbol{Z}}$ to the adversary. The adversary can now learn the second highest price sp by determining the highest non-zero position at $\boldsymbol{R} = \sum_{i=1}^{h}(\tilde{\boldsymbol{Z}}_i) + \sum_{j=1}^{k}(\boldsymbol{Z}'_j)$. So, the adversary learns nothing beyond the output (b^*, sp) at the end of the protocol.

□

$\mathcal{S}(\{\boldsymbol{x}'_j, \boldsymbol{r}'_j\}_{j\in[k]}, \{b'_j\}_{j\in[k]}, \textsf{out})$

1. Receive the output $\textsf{out} = (b^*, sp)$ from $\mathcal{F}$.
2. Sample random non-zero vectors $\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{r}}_i \in \mathbb{F}^n$ for $i \in [h]$. It also computes their corresponding commitments $\tilde{\mathtt{x}}_i, \tilde{\mathtt{r}}_i$ and validity proofs $\tilde{\pi}_{x_i}, \tilde{\pi}_{r_i}$.
3. Sample random bid b_i for $i \in [h]$ such that one of them is equal to b^* and others are smaller. Also, set one of the bids to sp such that others (excluding b^*) are smaller or equal.
4. Compute the polynomial commitment to unary encoding of the bids b_i for $i \in [h]$. It also computes their corresponding validity proofs $\tilde{\pi}_{b_i}$.
5. Knowing the description of AV, compute $\mathbf{Y} = \mathbf{M} \cdot \mathbf{X}$ including row-vector X'_i for $i \in [k]$ received from an adversary $\mathcal{A}$.
6. Compute vector $\boldsymbol{Z}_i = (\boldsymbol{x}_i + \boldsymbol{b}_i \circ \boldsymbol{r}_i) \circ \boldsymbol{Y}_i$ on behalf of honest bidders together with validity proofs $\tilde{\pi}_{Z_i}$ for $i \in [h]$.
7. Receive the values $\boldsymbol{Z}'_j$ for $j \in [k]$ from $\mathcal{A}$. If $\boldsymbol{Z}'$ is not of valid form, it aborts.
8. Compute the vector $\boldsymbol{R} = \sum_{i=1}^{h}(\boldsymbol{Z}_i) + \sum_{j=1}^{k}(\boldsymbol{Z}'_j)$. Note that $\mathcal{A}$ can also compute the same vector at this point.
9. The highest bid is highest position w such that $\boldsymbol{R}_w \neq 0$. Check if $R_w = b^*$; otherwise, it aborts.
10. Set $b^* = 0$ and compute its corresponding vector $\boldsymbol{Z} = (\boldsymbol{x} + \boldsymbol{b}^* \circ \boldsymbol{r}) \circ \boldsymbol{Y}$.
11. Compute the vector $\boldsymbol{R} = \sum_{i=1}^{h}(\boldsymbol{Z}_i) + \sum_{j=1}^{k}(\boldsymbol{Z}'_j)$.
12. The second highest bid is highest position w such that $\boldsymbol{R}_w \neq 0$. Check if $R_w = sp$; otherwise, it aborts.

Figure 8: Simulator for Cryptobazaar

B.2 Proof for Lemma 4

Proof. Completeness is immediately followed by writing out the verification equation as $e(G,H)^{f(\tau)-f(\gamma)+\gamma q(\gamma)} = e(G,H)^{\tau q(\tau)}$. So, this is just the equation $q(X) = \frac{f(X)-f(\gamma)}{X-\gamma}$ evaluated at a challenge point γ. Next, we proceed to show that this argument is knowledge sound by building an efficient extractor $\mathcal{E}$ that extracts the witness and then show how to build a simulator.

Soundness. Suppose that malicious $\mathcal{P}$ commits to $f(X), \mathbf{X}$ such that there exists i for which $X_i \neq f(w^i)$. Denote $f(\gamma) = y$. Then, except with negligible probability we have $\sum_{i=0}^{n} L_i(\gamma)\mathbf{X}_i = [y']_1 \neq [y]_1$. Finally, from soundness of the KZG commitment the malicious $\mathcal{P}$ has only negligible probability in constructing an accepting opening proof that $f(\gamma) = y' \neq y$.

Zero knowledge. We construct $\mathcal{S}_{\mathsf{pv}}$ such that given instance $\mathtt{x}, \mathbf{X}$, toxic waste τ, and verifier's challenge produces a transcript that is identically distributed to the transcript obtained from interaction with the honest prover that has a valid witness. Given γ, we have $[y]_1 \leftarrow \sum_{i=0}^{n} L_i(\gamma)\mathbf{X}_i$ that looks random given DDH assumption. Simulator then computes $\mathtt{q} = (\mathtt{f} - [y]_1)(\tau - \gamma)$ which is randomly distributed over γ. It is easy to check that $\mathtt{q}$ satisfies verifier's pairing check and therefore $\mathcal{S}_{pv}$ is able to produce a valid transcript which is indistinguishable from the one in the actual protocol. □

B.3 Proof for Lemma 5

Proof. We proceed to show this argument is knowledge sound by building an efficient extractor $\mathcal{E}$ that extracts the witness. Afterwards, we prove zero-knowledge by building up a simulator.

Knowledge soundness. As adversary $\mathcal{A}$ is algebraic, whenever it sends a commitment to some polynomial it also sends the actual polynomial. Further, if the verifier $\mathcal{V}$ does not accept then $\mathcal{A}$ clearly does not win the game, thus further assume that $\mathcal{V}$ accepts. Then, since $\mathcal{A}$ is algebraic, it sends $\hat{s}(X)$ and $q(X)$ together with $\mathtt{s}$ and $\mathtt{q}$. $\mathcal{E}$ then reconstructs $r(X)$ from $\hat{s}(X)$, $q(X)$, and (publicly-known) $z_H(X)$ in polynomial time.

Zero knowledge. We construct $\mathcal{S}_{nz}$ that, given instance $\mathtt{r}$, trapdoor τ and verifier randomness, produces a transcript that is equally distributed as the transcript obtained from the interaction with the honest prover that has a witness. Note that $\mathcal{S}_{nz}$ samples all values $\mathtt{s}, \mathtt{q}, r_\gamma, s_\gamma, q_\gamma$ uniformly at random. In the real execution $r(X)$ and $s(X)$ are blinded with $m(X)$ and $b(X)$, respectively, thus commitments and evaluations also look random. The simulator then computes $\mathtt{c} = \mathtt{r} + \alpha\mathtt{s} + \alpha^2\mathtt{q}$ and $y = r_\gamma + \alpha\hat{s}_\gamma + \alpha^2 q_\gamma$ as in the real protocol execution. Finally, by knowing trapdoor τ it computes $\mathtt{t} = (\mathtt{c} - [y]_1)(\tau - \gamma)$. □

B.4 Proof for Lemma 6

Proof. If $\boldsymbol{b}$ is a valid unary encoding, then it is straightforward to see that Δ has the defined structure. In particular, it consists of all zeros except a one at index i. From the other side, suppose that Δ has defined structure and that $\boldsymbol{b}$ starts with one. Let set index i such that $\Delta_i = 1$ and $i \neq n$. From this we have $\boldsymbol{b}_i - \boldsymbol{b}_{i+1} = 1$. Given that all the other values of Δ are zero we can derive that $\boldsymbol{b}_j = \boldsymbol{b}_{i+1}$ for $j \in [i+1, n]$ and that $\boldsymbol{b}_j = \boldsymbol{b}_i$ for $j \in [1, i]$. Since $\boldsymbol{b}_1 = 1$, thus $\boldsymbol{b}_j = 1$ for $j \in [1, i]$ and from this $\boldsymbol{b}_j = 0$ for $j \in [i+1, n]$, the claim is proved. In case that $i = l$, with the same reasoning we can conclude that $\Delta_n = \boldsymbol{b}_n = 1$ and thus for $j \in [1, n-1]$ we have $\boldsymbol{b}_j = \boldsymbol{b}_n = 1$. □

B.5 Proof for Lemma 7

Proof. We show that this argument is knowledge sound by building an efficient extractor $\mathcal{E}$ that extracts the witness and then proceed to build a simulator.

Knowledge soundness. If $\mathcal{V}$ does not accept then $\mathcal{A}$ clearly does not win the game, thus further assume that $\mathcal{V}$ accepts. By the knowledge soundness of the log derivative argument we know that if verifier accepts then evaluations of $f(X)$ indeed consists of one '1' and all zeroes except with negligible probability. Then from knowledge soundness of KZG we argue that $u(1) = 1$ except with negligible probability. Finally as $\mathcal{A}$ is algebraic, whenever it sends a commitment to some polynomial it also sends the actual polynomial. Therefore, $\mathcal{A}$ send polynomial $f(X)$. Then, $\mathcal{E}$ computes $u(w)$ from $f(1)$ and $u(1)$ and then it iteratively computes all $u(w^i)$ from $f(w^{i-1})$ and $u(w^{i-1})$.

Zero knowledge. We construct $\mathcal{S}_{\mathsf{unary}}$ such that given instance $\mathtt{u}$, toxic waste τ and verifier' challenge, produces a transcript that is identically distributed to the transcript obtained from interaction with honest prover that has a witness. Note that polynomials $u(X), f(X), B(X)$ are properly masked such that their commitments and openings have random distributions. $\mathcal{S}_{\mathsf{nz}}$ can then simply sample random elliptic curve and field elements and with knowledge of τ it can simulate correct opening proofs. Further, by the definition of zero-knowledge univariate sumcheck $R(X), Q(X)$ at round 3 are also random. $\square$

B.6 Proof for Lemma 8

Proof. Given that our protocol is closely similar to the generalized IPA introduced in Bunz et al. [13], we just give a high level intuition and refer the reader to [13] for more details. Completeness is straightforward. Given that we use the same form of blinding the zero knowledge can be proven with the identical method of [13]. Soundness can be proven as follows. The extractor first runs the extractor of $\mathcal{R}_{\mathsf{pse}}$ to extract a', r. Then in the similar fashion, the extractor builds a tree of transcripts by rewinding and recursively extracts vectors $\mathbf{a}_i \in \mathbb{F}^{\frac{n}{2^i}}$ for $i \in (\log n, \log n - 1, \dots, 0)$. $\square$

B.7 $\mathcal{R}_{\mathsf{pse}}$ with Blinders

We show how to deal with the aforementioned leakage when $a(X), b(X)$ are not properly blinded. Our solution to handle this is by committing to two blinding polynomials $s_1(X), s_2(X)$ and proving that this polynomials are multiples of $Z_H(X)$, thus $\sum_{h \in \mathbb{H}} s_i(h) = 0$, and they do not affect the sumcheck. Then, we prove the argument is knowledge sound and we provide a simulator to prove it is indeed a zero-knowledge argument of knowledge. The complete protocol is presented in Figure 9.

B.8 Proof for Lemma 9

Proof. Completeness is straightforward. We argue that our protocol is knowledge sound in the Algebraic Group Model by building an efficient extractor $\mathcal{E}$ that extracts the witness. *Knowledge soundness* is defined by a game involving an algebraic adversary $\mathcal{A}$ and an efficient extractor $\mathcal{E}$. Given the SP, adversary $\mathcal{A}$ produces the instances $\mathtt{a}, \mathtt{b}, P$ and produces an interactive argument for the verifier. Then, the goal of the extractor is to interact with $\mathcal{A}$ and at the end to output the witness $x, r, a(X), b(X)$. We say that $\mathcal{A}$ wins the game if $\mathcal{V}$ accepts and one of the following relations is false: $\mathtt{a} = [a(\tau)]_1, \mathtt{b} = [b(\tau)]_1, xG + rH = P, \sum_{h \in \mathbb{H}} a(h)b(h) = x$. The protocol has knowledge soundness if there is efficient $\mathcal{E}$ and that $\mathcal{A}$ cannot win the game except with negligible probability. If $\mathcal{V}$ does not accept then $\mathcal{A}$ clearly does not win the game, thus further assume that $\mathcal{V}$ accepts. As $\mathcal{A}$ is algebraic, whenever it sends a commitment to some polynomial it also sends the actual polynomial. Therefore, during the protocol execution $\mathcal{A}$ sends $\mathtt{s}_1, \mathtt{s}_2, \mathtt{B}, \mathtt{R}, \mathtt{q}, \mathtt{D}, \mathtt{Q_s}$ together with $s_1(X), s_2(X), B(X), R(X), q(X), D(X), Q_s(X)$. First, by the knowledge soundness of the proof of opening of Pedersen commitment there exists extractor $\mathcal{E}_{\mathsf{Pedersen}}$ such that except with a negligible probability it extracts x, p, r, s, where $z_1 = cx + p$ and $z_2 = cr + s$. Further, the passing of all the KZG checks implies that by the knowledge soundness of KZG and Schwartz-Zippel all polynomial identities hold except with negligible probability. Since zero knowledge sumcheck relation also passes we conclude that $\sum_{h \in H} B(h) + c(a(h) + s_1(h))(b(h) + s_2(h)) = z_1$, except with negligible probability. From the check that $Z_H(X)$ divides both $s_1(X)$ and $s_2(X)$ we know that they do not affect the sum since $Z_H(h) = 0, \forall h \in \mathbb{H}$, thus $\sum_{h \in \mathbb{H}} s_i(h) = 0$. Therefore, $\sum_{h \in H} B(h) + ca(h)b(h) = z_1$. Now suppose that $\sum_{h \in \mathbb{H}} a(h) \cdot b(h) = x' \neq x$ and suppose that $\mathcal{P}$ commits to $B(X)$ such that $\sum_{h \in \mathbb{H}} B(h) = p' \neq p$. Then, we have that $cx + p = cx' + p'$ which is true only if $c = \frac{p'-p}{x-x'}$, that happens with probability $\frac{1}{|\mathbb{F}|}$. Thus we have shown that if $\mathcal{V}$ accepts then $\sum_{h \in \mathbb{H}} a(h)b(h) = x$ holds with overwhelming probability. The last ambiguity we have to deal with is extracting $a(X), b(X)$. Even though $\mathcal{A}$ is algebraic it does not send $a(X), b(X)$ since commitments $\mathtt{a}, \mathtt{b}$ are already in the instance. However, $\mathcal{E}$ has the access to $s_1(X), s_2(X)$ and from as_γ, bs_γ it can compute a_γ and b_γ. Thus rewinding $\mathcal{A}$ n times and obtaining different opening challenges γ_i enable $\mathcal{E}$ to fully interpolate $a(X), b(X)$. From witness extended emulation [47] here we know that this $\mathcal{E}$ is still efficient and that it fails only with negligible probability.

Round 1 Prover:

1. Sample random p, s and compute $Q = pG + sH$.
2. Compute $b_0 = \frac{p}{n}$ and sample random $b_1, b_2, b_3, b_4, b_5, b_6$, then compute $B(X) = b_0 + b_1X + b_2X^2 + b_3X^3 + b_4X^4 + (b_5 + b_6X)z_H(X)$, observe that $\sum_{h \in \mathbb{H}} B(h) = p$.
3. Sample random r_0, r_1, r_2, r_3 and compute $s_1(X) = (r_0 + r_1X) \cdot Z_H(X)$ and $s_2(X) = (r_2 + r_3X + r_4X^2) \cdot Z_H(X)$.
4. Send $Q, \mathtt{B} = [B(\tau)]_1, \mathtt{s_1} = [s_1(\tau)]_1, \mathtt{s_2} = [s_2(\tau)]_1$.

Round 1 Verifier: Send random challenges c and α.
Round 2 Prover:

1. Compute $z_1 = cx + p$ and $z_2 = cr + s$.
2. Compute $R(X), q(X)$ such that $B(X) + c(a(X) + s_1(X))(b(X) + s_2(X)) = \frac{z_1}{|\mathbb{H}|} + XR(X) + q(X)z_H(X)$.
3. Compute $D(X) = R(X) \cdot X^{N-1-(n-2)}$ where N is the length of SP proving key.
4. Compute $Q_s(X)$ such that $s_1(X) + \alpha s_2(X) = Q_s(X)Z_H(X)$.
5. Send $z_1, z_2, \mathtt{R} = [R(\tau)]_1, \mathtt{q} = [q(\tau)]_1, \mathtt{D} = [D(\tau)]_1, \mathtt{Q_s} = [Q_s(\tau)]_1$.

Round 2 Verifier: Sample and send a random opening challenges $\gamma \in \mathbb{F}^* \setminus \mathbb{H}, \beta$.
Round 3 Prover: Send openings $as_\gamma = a(\gamma) + s_1(\gamma)$, $bs_\gamma = b(\gamma) + s_2(\gamma)$, $B_\gamma = B(\gamma)$, $R_\gamma = R(\gamma)$, $s_{1,\beta} = s_1(\beta)$, $s_{2,\beta} = s_2(\beta)$.
Round 3 Verifier: Send random separation challenge v.
Round 4 Prover:

1. Set $W(X) = (a(X) + s_1(X) + v(b(X) + s_2(X)) + v^2B(X) + v^3R(X) + v^4q(X)) \cdot (X - \gamma)^{-1}$.
2. Set $T(X) = (s_1(X) + vs_2(X) + v^2Q_s(X)) \cdot (X - \beta)^{-1}$
3. Send $\mathtt{W} = [W(\tau)]_1, \mathtt{T} = [T(\tau)]_1$.

Round 4 Verifier:

1. Check that $cP + Q = z_1G + z_2H$.
2. Compute $Q_{S,\beta} = (s_{1,\beta} + \alpha s_{2,\beta}) \cdot z_H(\beta)^{-1}$.
3. Assert $1 \leftarrow \mathsf{KZG.Verify}(\mathtt{s_1} + v\mathtt{s_2} + v^2\mathtt{Q_s}, \beta, s_{1,\beta} + vs_{2,\beta} + v^2Q_{S,\beta}, q_T)$.
4. Compute $q_\gamma = (B_\gamma + ca_\gamma b_\gamma - \frac{z_1}{n} - \gamma R_\gamma) \cdot z_H(\gamma)^{-1}$.
5. Set $C = \mathtt{a} + \mathtt{s_1} + v(\mathtt{B} + \mathtt{s_2}) + v^2\mathtt{B} + v^3\mathtt{R} + v^4\mathtt{q}$.
6. Compute $y = as_\gamma + vbs_\gamma + v^2B_\gamma + v^3R_\gamma + v^4q_\gamma$.
7. Assert $1 = \mathsf{KZG.Verify}(\mathsf{SP}, C, \gamma, y, \mathtt{W})$.
8. Set degree bound $d = N - 1 - (n-2)$, where N is the length of SP proving key.
9. Assert $e(\mathtt{R}, [x^d]_2) = e(\mathtt{D}, [1]_2)$.

Figure 9: Interactive zero-knowledge argument of knowledge protocol Π_{pse} with blinders for relation $\mathcal{R}_{\mathsf{pse}}$.

To prove *zero knowledge* property, we construct $\mathcal{S}_{\mathsf{pse}}$ such that given instance $P, \mathtt{a}, \mathtt{b}$, toxic waste τ and verifier's challenge, produces a transcript that is identically distributed to the transcript obtained from interaction with honest prover that has a witness. In Figure 10, we outline the steps of the simulator and then argue that simulator produces an accepting transcript and that all messages are correctly distributed. It can be seen that all KZG checks are passing and that by definition $cX + Q = z_1G + z_2H$, therefore $\mathcal{V}$ accepts. We now argue about the indistinguishability of the real and simulated transcripts.

$\mathcal{S}_{\mathsf{pse}}((G,H), P, \mathtt{a}, \mathtt{b}, \tau, c, \gamma, v)$

1. Sample random z_1, z_2 and compute $Q = z_1G + z_2H - cX$.
2. Sample random a_1, a_2, a_3 and send $Q, \mathtt{B} = [a_1]_1, \mathtt{a_2} = [a_2]_1, \mathtt{a_3} = [a_3]_1$.
3. Sample random a_4, a_5, a_6 and send z_1, z_2, $\mathtt{R} = [a_4]_1$, $\mathtt{q} = [a_5]_1$, $\mathtt{D} = \tau^{N-1-(n-2)}[a_4]_1$, and $\mathtt{Q_s} = [a_6]_1$.
4. Sample random $B_\gamma, as_\gamma, bs_\gamma, R_\gamma, s_{1,\beta}, s_{s,\beta}$ and send them together with $q_\gamma, q_{S,\beta}$ such that
$$B_\gamma + ca_\gamma b_\gamma = \frac{z_1}{n} + \gamma R_\gamma + q_\gamma Z_H(\gamma)$$
and
$$Q_{S,\beta} z_H(\beta) = s_{1,\beta} + \alpha s_{2,\beta} \ .$$
5. Compute
   (a) $q_1 = (\tau - \gamma)^{-1} \cdot (q_a + q_{s_1} - [as_\gamma]_1)$
   (b) $q_2 = (\tau - \gamma)^{-1} \cdot (q_b + q_{s_2} - [bs_\gamma]_1)$
   (c) $q_3 = (\tau - \gamma)^{-1} \cdot (q_B - [B_\gamma]_1)$
   (d) $q_4 = (\tau - \gamma)^{-1} \cdot (q_R - [R_\gamma]_1)$
   (e) $q_5 = (\tau - \gamma)^{-1} \cdot (q_q - [q_\gamma]_1)$
   (f) $q_6 = (\tau - \beta)^{-1} \cdot (q_{s_1} - [s_{1,\beta}]_1)$
   (g) $q_7 = (\tau - \beta)^{-1} \cdot (q_{s_2} - [s_{2,\beta}]_1)$
   (h) $q_8 = (\tau - \beta)^{-1} \cdot (q_S - [Q_{S,\beta}]_1)$.
6. Compute and send $q_W = q_1 + vq_2 + v^2q_3 + v^3q_4 + v^4q_5$ and $q_T = q_6 + vq_7 + v^2q_8$.

Figure 10: Simulator for $\mathcal{R}_{\mathsf{pse}}$

1. z_1, z_2 are uniformly sampled, thus Q matches the actual distribution.
2. $\mathtt{s_1}, \mathtt{s_2}$ are blinded with r_0, r_2 for prover and a_2, a_3 for simulator respectively.
3. $\mathtt{B}$ is blinded with b_1 for prover and a_1 for simulator.
4. $\mathtt{R}$ is blinded with b_2 for prover and a_4 for simulator.
5. $\mathtt{q}$ is blinded with b_5 for prover and a_5 for simulator.

6. $\mathtt{Q_s}$ is blinded with r_4 for prover and a_6 for simulator.

7. B_γ is blinded with b_3 for prover and uniformly sampled for simulator.

8. R_γ is blinded with b_4 for prover and uniformly sampled for simulator.

9. q_γ is blinded with b_6 for prover and uniformly sampled for simulator.

10. $s_{1,\beta}, s_{2,\beta}$ are blinded with r_1, r_3 for prover and uniformly sampled for simulator.

11. as_γ is uniformly distributed since it has random contribution from both $a(X)$ and $s_1(X)$ for prover and it is uniformly sampled for simulator.

12. bs_γ is uniformly distributed since it has random contribution from both $b(X)$ and $s_2(X)$ for prover and it is uniformly sampled for simulator.

13. $\mathtt{W}, \mathtt{T}$ uniquely satisfy the KZG openings.

$\square$