Cryptobazaar: Private Sealed-bid Auctions at Scale

Andrija Novakovic[*]
Bain Capital Crypto
anovakovic@baincapital.com

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

Kobi Gurkan
Bain Capital Crypto
kgurkan@baincapital.com

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

***Abstract*—This work introduces Cryptobazaar, a 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 present protocol variants 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 auction run 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.**

I. 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 [57], for allocating block space and determining the transaction order to mitigate maximal extractable value (MEV) in cryptocurrencies [23], [49], or for renting out hardware to run large-scale computations or to store data [37], and more.

A digital auction system should provide various essential properties, including: (1) *privacy* for (losing) bidders to prevent disclosure of their bid preferences [37] which is particularly crucial to ensure fairness in iterative/sequential auctions, as any leakage may allow bidders to adapt their strategies in subsequent runs [23], [2], [32]; (2) *verifiability* to ensure that third parties (auditors) can verify the outcome and penalize misbehaving participants [32]; (3) *trust minimization* to mitigate the influence that a potentially malicious auctioneer could have on the auction's outcome [53]; 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) [37].

[*]Andrija Novakovic and Alireza Kavousi jointly led the research and contributed equally to this work.

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.240481
www.ndss-symposium.org

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 [54]. 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 [51], [28], [59] but none have managed to address the privacy issue as well so far. Other approaches rely on elaborated cryptographic machinery such as generic secure multi-party computation (MPC) [9] or fully homomorphic encryption (FHE) [33] 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 [60] 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 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 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 $(p+1)$st-price auctions (*i.e.,* uniform auctions) as well as an iterative/sequential mode with only a slight adjustment of the main protocol.

Technical overview. At its core, Cryptobazaar uses the *anonymous veto (AV)* protocol [31] to compute, in a scalable way, a distributed Boolean-OR over all unary-encoded bids to determine the highest bid, and thus the result of an auction run, while protecting the confidentiality of the losing bids. To ensure the well-formedness of all submitted values and maintain 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 Pedersen 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* [10], [14], *log-derivatives* [30], and *univariate sumcheck* [7] may be of independent interest, *e.g.,* to enhance verifiable shuffling [4]. In the event of having multiple candidate winners, we break possible ties via public randomness [35] 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 without re-running the protocol which is common practice for private Vickery auctions [40]. We propose a variant to run secure iterative/sequential auctions using setup re-randomization techniques, making Cryptobazaar particularly appealing for practical use cases. Finally, we consider real-world deployment with a focus on Ethereum and discuss how Cryptobazaar can be used as an alternative auction protocol in their block building process [23]. Our systems' performance easily satisfies Ethereum's requirement that the auction terminates in a fraction of its 12-second block time window. We provide a graphical overview of Cryptobazaar's protocol variants in Figure 1.

Contributions. In this work, we show how to build a private, scalable, and trustless sealed-bid auction protocol, called Cryptobazaar, by combining our variant of an anonymous veto (AV) protocol over unary-encoded bids with various novel succinct zero-knowledge proof gadgets. Our new system improves over previous state-of-the-art private auction protocols that either are not scalable [3], or require stronger trust assumptions such as a threshold [47] or anytrust [60] model. In summary, we make the following contributions:

- We propose Cryptobazaar, a private scalable sealed-bid auction protocol that supports first- and second-price auctions and relies only on an untrusted auctioneer for coordination, a critical improvement over the state-of-the-art [3], [60].
- We co-design various novel (public coin) zero-knowledge succinct arguments of knowledge to ensure soundness and privacy of the different protocol steps.
- We present two extensions to Cryptobazaar showing how to run uniform auctions and sequential/iterative auctions efficiently without having to re-execute the full protocol.
- We demonstrate the practicality of our system by evaluating an open-source implementation of Cryptobazaar in Rust. A run of the auction with 128 bidders and a price range of 1024 values terminates in less than 0.5 second and requires each bidder to communicate only about $32\,\text{KB}$ of data.

II. Background

A. Notation

We write $x \xleftarrow{\$} S$ to denote uniform sampling of element x from finite set S and $[n]$ for the set of integers $\{1, \ldots, n\}$. 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.,* $\boldsymbol{a} = (a_1, \ldots, a_n) \in \mathbb{F}^n$. 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 $\boldsymbol{a} \circ \boldsymbol{b} = (a_1 \cdot b_1, \ldots, a_n \cdot b_n)$. 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 $a \cdot \boldsymbol{G} = (a \cdot G_1, \ldots, a \cdot G_n)$. Given $1 \leq l \leq n$ and vector $\boldsymbol{a} \in \mathbb{F}^n$, we denote the slice of the first l elements of $\boldsymbol{a}$ by $\boldsymbol{a}_{[:l]} = (a_1, \ldots, a_l) \in \mathbb{F}^l$ and the slice of the last $n-l$ elements of $\boldsymbol{a}$ by $\boldsymbol{a}_{[l:]} = (a_{l+1}, \ldots, a_n) \in \mathbb{F}^{n-l}$. 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 $n = \text{poly}(\lambda)$.

B. 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.

C. 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* [44] is an example of a widely-used commitment scheme that is perfectly hiding and

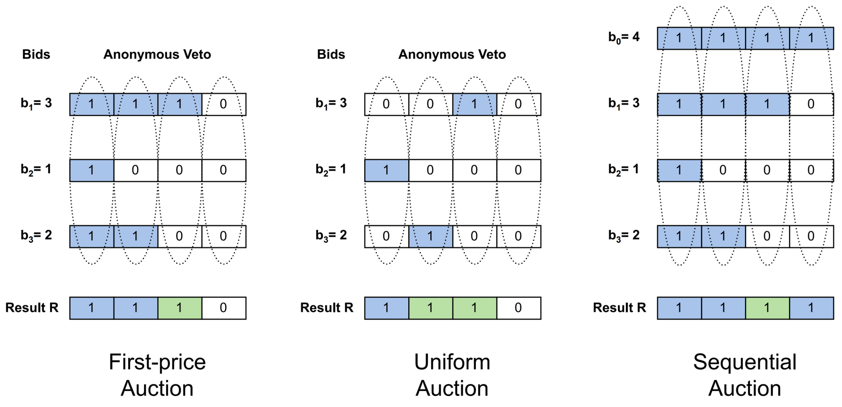

Fig. 1: Example executions of different Cryptobazaar protocol variants. Dotted ellipses resemble the anonymous veto (AV) instances running on bids b_i producing result bit strings R. First-price and sequential variants use a unary encoding whereas the uniform variant uses a Boolean encoding. Entries in R deciding on winners are highlighted in green. Whereas, first-price and sequential auctions have only one winner, uniform auctions can have multiple ones and sell multiple items (here two) in one protocol run. Sequential auctions use an additional masking all-1 bid (here b_0) submitted by the auctioneer to avoid some privacy-leakage. We omit the illustration of second-price auctions from this overview to avoid any confusion and refer the reader instead to the text.

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* [15] 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)* [34] 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 it should hold $\mathsf{PCS.Verify}(\mathsf{SP}, C, w, \phi(w), \pi) = 1$ 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 polynomial commitment scheme [34]:

- $\mathsf{SP} \leftarrow \mathsf{KZG.Setup}(1^\lambda, d)$: Outputs a bilinear pairing group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, G, H, e)$, and system parameters $\mathsf{SP} = [\tau^i]_1 = (G, \tau G, \dots, \tau^d G)$ 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* [58] 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 [42]. Alternatively, one could use a PCS that does not require a trusted setup at the cost of a performance trade-off [6].

D. Anonymous Veto

In Cryptobazaar we use the *anonymous veto (AV)* protocol [31] 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. It works as follows:

- Each party i samples a random value $x_i \in \mathbb{F}$, and broadcasts x_iG. After seeing messages from all other parties, i computes $Y_i = \sum_{j=1}^{i-1} x_jG - \sum_{j=i+1}^{n} x_jG$.
- After obtaining Y_i, each party i computes a value R_i as either $R_i = r_iY_i$ if they veto or $R_i = x_iY_i$ 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 $\sum_i x_iY_i = 0$ (see Proposition 1, [31]). To illustrate this equality more intuitively, let $\boldsymbol{X}$ be the vector of elements obtained after the first AV round, *i.e.,* $\boldsymbol{X}_i = x_iG$, 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 $\langle \boldsymbol{X}, \mathbf{M} \cdot \boldsymbol{X}^T \rangle = 0$. 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_iY_i and x_iY_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.

E. Zero-knowledge Argument of Knowledge

In Cryptobazaar we deploy *zero-knowledge succinct non-interactive arguments of knowledge (zkSNARKs)* [43], [50] to enforce honest behavior of the parties and ensure public verifiability. We refer the reader to the cited literature for a comprehensive survey and formal definitions on zkSNARKs.

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 [22]. 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)* [26]. There are different ways to build zkSNARKs concretely. We follow the approach in *PLONK* [27] by taking polynomial *interactive oracle proofs* (IOP) [8], 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.

Inner-product Arguments (IPA). An inner-product argument [10] is an efficient argument system for the following relation:

$$\mathcal{R} = \left\{ \begin{pmatrix} \boldsymbol{G}, \boldsymbol{H} \in \mathbb{G}^n, P \in \mathbb{G}, \\ c \in \mathbb{F}_p; \boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}_p^n \end{pmatrix} : \begin{matrix} P = \langle \boldsymbol{a}, \boldsymbol{G} \rangle + \langle \boldsymbol{b}, \boldsymbol{H} \rangle, \\ c = \langle \boldsymbol{a}, \boldsymbol{b} \rangle \end{matrix} \right\}$$

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. [13] 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 [14].

Univariate Sumcheck Protocol. The univariate sumcheck protocol [7] 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 [61].

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.

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 [30].

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, Haböck [30] 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 $z_1, \ldots, z_n$.

III. Cryptobazaar

In this section we present Cryptobazaar, our private scalable sealed-bid auction protocol. Section III-A discusses system and threat models, Section III-B presents our main design goals, Section A provides formal auction definitions, and Section III-C discusses the protocol details.

A. 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 information about the number of protocol participants or their identities. Our communication pattern resembles a star topology with the auctioneer being the coordinator. 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 auctioneers are usually present in practice [60]. We model bidders and auctioneer as malicious adversaries who may arbitrarily deviate from the correct execution of the protocol. As required in a private auction [60], we implicitely assume all the parties involved are authorized so that they can be subject to penalties (*e.g.,* financial or reputational). We assume that authentication between bidders and auctioneer and the penalizing of misbehaving parties is handled out of band and discuss this further in Section V. The auditor gets involved at the end of the auction to validate the outcome using the information posted on the public log.

B. Design Goals

Cryptobazaar aims to achieve the following design goals:

- **Privacy:** The protocol discloses the first and second highest bids 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 decentralized 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 computation and bandwidth overheads for bidders and auctioneer.

We provide the formal security definitions in Appendix A.

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:

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 II-D.
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 $(\boldsymbol{Z}_i, \pi_{Z_i})$ to the public log.

Once all bidders are done, the auctioneer executes:

4) Compute vector $\boldsymbol{R}$ s.t. $\boldsymbol{R}_j = \sum_{i=1}^{m}(\boldsymbol{Z}_i)_j$ 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.

Fig. 2: The Cryptobazaar protocol

C. Protocol Description

Figure 2 presents an overview of Cryptobazaar which consists of the two phases of *preprocessing*[1] and *bidding*. Below we focus on the core protocol and present the necessary validity proofs only abstractly. We defer the discussion of the proofs' details to Section IV.

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 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

[1]This should not be confused with the similar notion in the MPC literature where the inputs are not available, although the computation here is merely on random values.

II-D), 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 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 $\boldsymbol{R} = \sum_{i=1}^{m}(\boldsymbol{Z}_i)$. The highest index w of a 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 [35] to choose among the set of candidate winners.

For a first-price auction, we are done at this point. In Section V, we discuss how to run a second-price auction and further propose a variant that efficiently supports second-price and more generally $(p+1)$st-price (*i.e.,* uniform) auctions [36] without having to re-run the protocol.

Theorem 1. The Cryptobazaar protocol (Figure 2) achieves completeness, soundness, and privacy.

We provide the proof to Theorem 1 in Appendix A1.

IV. Validity Proofs

In this section we present the technical details of the validity proofs used in Cryptobazaar. In Section IV-A 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 IV-B we discuss the bidding phase proofs π_{Z_i} (Step 2) and π_w (Step 6).

A. 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)[2] of $\boldsymbol{x}$. We provide the corresponding argument Π_{pv} in Figure 3 which proves

[2]It refers to the process of extending a function defined on a small domain (like a finite field subset) to a larger domain as a low-degree polynomial.

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 $e(\mathtt{x} - [y]_1 + \gamma\mathsf{q}, [1]_2) = e(\mathsf{q}, [\tau]_2)$.

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

that $\boldsymbol{X}$ and polynomial commitment $\mathtt{x}$ encode the same vector $\boldsymbol{x}$. The intuition is as follows: After receiving the challenge γ from the verifier, the prover computes and sends a KZG opening proof q for the evaluation of f at γ back to the verifier. In the second round, instead of obtaining an evaluation claim from the prover, the verifier computes $[y]_1$ as a multi-scalar multiplication (MSM) between the committed values $\mathbf{X}$ and the Lagrange basis evaluations $L_i(\gamma)$. It then uses $[y]_1$ to check the given pairing equation via a standard KZG opening check to verify the overall claim.

Lemma 4. The protocol Π_{pv} for relation $\mathcal{R}_{\mathsf{pv}}$, see Figure 3, satisfies completeness, soundness, and zero-knowledge.

We provide the proof for Lemma 4 in Appendix A2.

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 $\mathbb{H}$ [34]. 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 [16] 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 [16]. Further, the use of vanishing polynomial makes the evaluations of $\hat{f}(X)$ be equal to those

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 $\mathsf{s} = [\hat{s}(\tau)]_1, \mathsf{q} = [q(\tau)]_1$.

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 $(\mathtt{t}, \cdot) = \mathsf{KZG.Open}(\mathsf{SP}, \phi(X), \gamma)$.
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 $\mathsf{c} = \mathtt{r} + \alpha\mathsf{s} + \alpha^2\mathsf{q}$.
3) Assert $1 = \mathsf{KZG.Verify}(\mathsf{SP}, \mathsf{c}, \gamma, y, \mathtt{t})$.

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

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 $\mathcal{R}_{nz}$ showing that $r(w^i) \neq 0$ for each $w^i \in \mathbb{H}$.

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

We present the corresponding argument Π_{nz} in Figure 4. The intuition is as follows: The prover starts by sampling blinding polynomials $b(X)$ and $m(X)$, to setup blinded polynomials $\hat{s}(X)$ and $\hat{r}(X)$, respectively, for the computation of the KZG polynomial $q(X)$ which encodes that $r(X)$ is invertible over $\mathbb{H}$ (round 1, step 5). The blinding process ensures that the KZG opening proofs s and q (round 1, step 6) do not leak information about the underlying bids. Afterwards, prover and verifier engage with each other to batch KZG opening proofs to reduce the proof size and the number of pairings required for verification [27]. The verifier initiates this process by sending a random separation challenge α in round 2 and ends it by checking the batched KZG opening proof in round 3, step 3.

Lemma 5. The protocol Π_{nz} for relation $\mathcal{R}_{\mathsf{nz}}$, see Figure 4, satisfies completeness, soundness, and zero-knowledge.

We provide the proof for Lemma 5 in Appendix A3.

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 II-E, 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 A4.

To prove the correct form of Δ, we use logarithmic derivatives [30]. Due to the unique fractional decomposition of logarithmic derivatives, see Section II-E, 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 [3] 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:

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

We present the corresponding argument Π_{unary} in Figure 5. The intuition is as follows: In round 1, the prover computes the difference polynomial $f(X)$ and uses it to implicitly show that $u(X)$ extends a valid unary encoding, as stated in Lemma 6. The whole process is blinded to ensure zero-knowledge. In round 2, the prover interpolates a polynomial $B(X)$ that is used in a log derivative proof, and it computes a witness polynomial $Q_B(X)$ to show the well-formedness of $B(X)$. In round 3, the prover computes witness polynomials $R(X)$ and $Q(X)$ to prove the univariate sumcheck relation. In round 4, the verifier sends a batching challenge v so that the prover can aggregate all KZG instances. The verifier then checks if $u(1) = 1$ in the final round as required by Lemma 6. Finally, in round 5, the verifier checks the log derivative relation and the batched KZG opening proof.

Lemma 7. The protocol Π_{unary} for relation $\mathcal{R}_{\mathsf{unary}}$, see Figure 5, satisfies completeness, soundness, and zero-knowledge.

[3]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 [50] for more details.

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 $S(X) + \alpha\hat{B}(X) = \frac{s+\alpha\cdot v}{n} + XR(X) + Q(X)z_H(X)$.
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 $A(X) = (u(X) + vf(X) + v^2Q_f(X) + v^3\hat{B}(X) + v^4S(X) + v^5Q_B(X) + v^6R(X) + v^7Q(X)) \cdot (X+\beta)^{-1}$.
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 $y = u_\beta + vf_\beta + v^2Q_{f,\beta} + v^3B_\beta + v^4s_\beta + v^5Q_{B,\beta} + v^6R_\beta + v^7Q_\beta$.
5) Assert that $1 = \mathsf{KZG.Verify}(\mathsf{SP}, C, \beta, y, \mathtt{a})$ and $1 = \mathsf{KZG.Verify}(\mathsf{SP}, \mathtt{u}, w\beta, u_{w,\beta}, \mathtt{a_w})$.

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

Verifier: Send a random scalar c.

Prover and Verifier:

1) Compute $\mathbf{c} = (1, c, c^2, \ldots, c^{n-1})$ 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 $L + r_lH$ and $R + r_rH$.

Verifier:

4) Sample and send a random challenge α.
5) Compute $C' = \alpha^{-1}L + C' + \alpha R$.

Prover:

6) Compute $C' = \alpha^{-1}L + C' + \alpha R$.
7) Set $a' = \langle \mathbf{a}'_{lo}, {\mathbf{a}'_{hi}}^{\alpha} \rangle$ and $X' = \langle \mathbf{X}'_{lo}, {\mathbf{X}'_{hi}}^{\alpha^{-1}} \rangle$.
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 $\mathcal{R}_{\mathsf{pse}}((\mathbf{pp}, \mathbf{srs}, X', H); (q_a, \tilde{f}(X), C'); (a(X), a', r))$.

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

We provide the proof for Lemma 7 in Appendix A5.

B. 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 $\boldsymbol{a} = \boldsymbol{x} + \boldsymbol{b} \circ \boldsymbol{r}$. To ensure $\boldsymbol{a} = \boldsymbol{x} + \boldsymbol{b} \circ \boldsymbol{r}$, each bidder runs a check of polynomial equalities at a random point which is a standard technique [50] and we thus do not make it explicit below. We prove $\boldsymbol{Z} = \boldsymbol{a}\boldsymbol{Y}$ by 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. This random linear combination of terms guarantees that the equivalency holds for all $j \in [n]$ and no terms cancel out. 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 [14] between $\boldsymbol{a}$ and $\boldsymbol{Y}$. Thus, we need to develop an argument of knowledge for the following relation:

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

We present the corresponding argument Π_{sum} in Figure 6. The intuition is as follows: The protocol is similar to a generalized inner-product argument [14]. At each round, the prover folds its witness vectors by half, given a random verifier challenge α. At the end, the verifier obtains a link between a Pedersen commitment (*i.e.,* masked vector of length 1) and a univariate sumcheck claim. Prover and verifier then engage with each other in proving the relation $\mathcal{R}_{\mathsf{pse}}$.

Lemma 8. The protocol Π_{sum} for relation $\mathcal{R}_{\mathsf{sum}}$, see Figure 6, satisfies completeness, soundness, and zero-knowledge.

We provide the proof for Lemma 8 in Appendix A6.

Removal of one inner product. As part of the argument 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 [7]. 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 for F and C which matters security-wise, as we need to use the same set of challenges for both [24]. The former is needed to convince $\mathcal{V}$ that $\mathcal{P}$ is indeed using $\boldsymbol{a}$. Following the IPA protocol [13], [14], 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 [12], [14].[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 [44] and univariate sumcheck [7] that may be of independent interest. Given a Pedersen Commitment $P = xG + rH$, the 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:

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

We present a zero-knowledge argument realizing the relation $\mathcal{R}_{\mathsf{pse}}$ in Figure 7. At a high level, there are two protocols that run in parallel: one for proving knowledge of a Pedersen commitment and one for a zero-knowledge univariate sumcheck. 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 protocol we notice that in our case $a(x)$ is properly blinded and $b(x)$ is a public

[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.

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 $e(\mathtt{R}, [x^d]_2) = e(\mathtt{D}, [1]_2)$.

Fig. 7: Interactive zero-knowledge argument of knowledge protocol Π_{pse} for relation $\mathcal{R}_{\mathsf{pse}}$.

polynomial, guaranteeing zero-knowledge. So, we state the relation assuming the openings of $a(X), b(X)$ do not affect zero-knowledge. That said, we present a generalized argument to cover the situations where the witness polynomials are committed without blinders. We handle this by committing two blinding polynomials $s_1(X), s_2(X)$ and proving that these polynomials are multiples of $Z_H(X)$ and they do not affect the sumcheck, thus $\sum_{h \in \mathbb{H}} s_i(h) = 0$. In the last round, the prover batches all openings. The verifier checks the Pedersen opening and all KZG openings and runs one pairing-based degree check and one KZG verification to check the univariate sumcheck relation. 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 7 satisfies completeness, soundness, and zero-knowledge.

We provide the proof for Lemma 9 in Appendix A7.

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 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) $\boldsymbol{Z} = \boldsymbol{a}\boldsymbol{Y}$. 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 [48] 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 [48] 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 [35] 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 [5].

V. Extensions and Deployment Considerations

In this section we present two extensions to our main protocol and discuss considerations for practical deployments.

A. Efficient Second-price Auctions

Our main protocol in Figure 2 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 [60]. A caveat is that if multiple bidders collude to bid the maximum but remain unresponsive, the protocol may not terminate as there is no way to exclude all the colluding parties. We now show how to modify the $\mathsf{Cryptobazaar}$ 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 $w' < w - p + 1$.

In case multiple bids are placed at the same price we choose one with public randomness similar to the main protocol. 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 [19]) and could be of interest depending on the applications, *e.g.,* when moving from single-item to multi-item NFT auctions [39].

B. 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 observed in Addax [60]. 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 [23]. 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 [21].

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 [52].

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 set membership proof), the winner is incentivized to re-do the preprocessing phase to protect its privacy for future runs of the auction.

C. Non-Responsive Participants

An inherent challenge of commit-reveal style protocols in a practical deployment is the risk of participants aborting during the protocol execution. Since Cryptobazaar implements a private commit-reveal variant, it is also susceptible to this issue which we address by incorporating slashing, a widely-used solution in practice, to disincentivize parties from misbehaving. Further, due to the privacy properties of the AV protocol, there is no way to determine the winner if they do not come forward voluntarily. Although winners have no incentive to not claim their wins, we recommend to include a timeout until which the highest bidder can claim their win. If they do not, the second-highest bidder gets a chance. This process then continues until some bidder presents a valid proof for the currently eligible slot ensuring that the protocol terminates eventually.

VI. Evaluation

We implemented Cryptobazaar in Rust using the cryptography framework `arkworks` [62].[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 multi-scalar 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` [45] or `gnark` [11] 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 IIIa.

Note that the computation of the validity proofs is not on the critical path for certain deployment scenarios, like running an auction via optimistic roll-ups [46].

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 IIIb. 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 IIIc. 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.

Reducing Computational Overheads. The original AV protocol has linear $O(n)$ communication cost (assuming broadcast) and quadratic $O(n^2)$ computational cost. 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) [20] of size n between the first row of $\mathbf{M}$ and $\boldsymbol{X}$.

Practical deployment considerations. To shed some more light on the practical usefulness of Cryptobazaar, we take a closer look at Ethereum and its proposer-builder separation (PBS) architecture [25] which heavily relies on auctions. A practically efficient sealed-bid auction would reduce the trust assumptions on the relay (as auctioneer) while preventing intense competition among builders (as bidders). In the current realization [23], 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. This approach reveals the complete bid vector to the relay, though, thereby leaking information about builders' bids and bidding strategies. A more privacy-friendly mechanism would minimize this leakage and reveal only the winning builder and its corresponding bid. However, simply eliminating the relay and allowing all builders to submit bids directly to the proposer is not viable: such a naive design would create a denial-of-service vulnerability, as adversaries could flood proposers with spurious bids from many fake "builders" and thereby disrupt block production. Any relay-free or improved PBS design must therefore simultaneously address the challenges around bid privacy and DoS resilience. Cryptobazaar provides such a solution where a trustless coordinator, which can be the relay or the proposer itself, verifiably computes the winning bid in a privacy-preserving and DoS-resilient manner. According to recent a analysis [56], the block preparation is dominated by 25 builders ([56], Table 5). Moreover, the authors collected 191 builder public keys ([56], Table 2), which means that 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). Thus, our experiments with 32, 128, and 256 bidders reflect well what currently deployed large-scale systems like Ethereum require. When a validator is chosen to propose a block, it has only a few seconds to find the auction winner, and broadcast the block. Thus, ensuring a low latency execution for the auction protocol is critical. Our basic protocol variant with 128 bidders and a price range of 1024 values terminates in less than 0.5 second easily satisfying the latency requirement. If needed this could be further reduced to a few hundred milliseconds in an optimistic mode (with asynchronous proof generation). Furthermore, bids range usually from cents to

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

TABLE I: 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 II: 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 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 trust assumptions or not. Versatility captures the ability to (securely) support different protocol variants without major modifications.

Protocols	Privacy	Scalability	Trust minimization	Versatility
Riggs [51]	○	◐	●	○
Cicada [28]	○	◐	●	○
SEAL [3]	●	○	●	○
Addax [60]	●	●	○	◐
Cryptobazaar	●	●	●	●

tens of dollars and a realistic deployment would likely have a price range of $1000 \leq n \leq 10000$ depending on the application [60]. We further remark that services like Flashbots that offer sealed-bid auctions typically have *off-chain* bidding processes (*e.g.,* submitting bids, determining winners, etc.) on their own servers and only the actual transaction execution happens on-chain [1]. Finally, the material for an auction run such as validity proofs need not to be stored permanently and could be deleted after some time, *e.g.,* once auditing is done, reducing overall storage requirements.

VII. Related Work

Riggs [51] 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. This work particularly addresses the practical details of the deployment setting [18], [55] including running auctions in parallel while locking up enough collateral without privacy leakage. Cicada [28] is another recent auction protocol that differs from Riggs by proposing a non-interactive protocol via homomorphic time lock puzzles [38] 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, making them unsuitable for an iterative auction mode due to the leakage of the bids in each run. 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. Since an auction can be considered an evaluation of a function of the bids, a natural approach to achieve (full) privacy is to encode the auction as a generic MPC among the bidders [41]. Despite recent advances in efficiency of generic MPC, it is still impractical when there are many bidders due the the inherent need for rounds of communication among them. An alternative is to use a delegated MPC setting whereby two parties (*i.e.,* non-colluding auctioneers) run the MPC on behalf of others, with bidders sending the shares of their bids. However, this setting lacks trust minimization and also integrity: either party is free to provide bogus to the MPC causing the result of the auction's run to be undetectably incorrect [60]. With Cryptobazaar, we can efficiently support auction variants with a practical system design (coordinated by an untrusted auctioneer), and public verifiability to ensure the integrity of all protocol steps.

One of the early works on private auction is due to Sako [47]. It associates each admissible price with a distinct (threshold) public encryption key. Each bidder then submits a ciphertext encrypted under the key corresponding to their chosen price. During the opening phase, a committe of auctioneers trial decrypt all of the submitted ciphertexts starting with the decryption key of the highest price and stop as soon as there is at least one ciphertext on a given level that decrypts successfully. Thereby the auctioneers only learn the maximum bid and the bidders who placed it. However, Sako's protocol relies on a threshold security assumption whereas Cryptobazaar works with minimal trust assumptions. Two recent stand-alone and private auction protocols are SEAL [3], and Addax [60]. 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 [31] 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 must remain online during the auction to dynamically update their inputs to ensure correctness (*i.e.,* , that the output reflects the highest bid), limiting the usability of the system. A major issue of this approach is that whenever a bidder realizes they are not the winner (because another bid is higher), they must voluntarily adjust their remaining inputs, so that the final output corresponds to the winning bid. Moreover, SEAL uses traditional Sigma protocols instead of more efficient succinct validity arguments. Addax [60] 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 [17] allowing an auction to be conducted over secret-shared bids. Cryptobazaar shares some common properties with Addax including using price range 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, particularly Affine-aggregatable encodings (AFE). Although a side-by-side comparison between Cryptobazaar and Addax is difficult due to the different settings (*e.g.,* threat models), we provide some concrete 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 and Rex Fernando for constructive discussions on auction theory. This work was partially done while Andrija Novakovic and Kobi Gurkan were working at Geometry Research. This work has been supported by a grant from the Ecosystem Support Program of the Ethereum Foundation (ID FY24-1539).

References

[1] *Flashbots*, 2022.

[2] R. Alvarez and M. Nojoumian, *Comprehensive survey on privacy-preserving protocols for sealed-bid auctions*, Computers & Security, 88 (2020), p. 101502.

[3] S. Bag, F. Hao, S. F. Shahandashti, and I. G. Ray, *Seal: Sealed-bid auction without auctioneers*, IEEE Transactions on Information Forensics and Security, 15 (2019), pp. 2042–2052.

[4] S. Bayer and J. Groth, *Efficient zero-knowledge argument for correctness of a shuffle*, in Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2012, pp. 263–280.

[5] J. Bearer, B. Bünz, P. Camacho, B. Chen, E. Davidson, B. Fisch, B. Fish, G. Gutoski, F. Krell, C. Lin, et al., *The espresso sequencing network: Hotshot consensus, tiramisu data-availability, and builder-exchange*, Cryptology ePrint Archive, (2024).

[6] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. 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.

[7] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. 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, Springer, 2019, pp. 103–128.

[8] E. Ben-Sasson, A. Chiesa, and N. 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, Springer, 2016, pp. 31–60.

[9] E.-O. Blass and F. 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, Springer, 2018, pp. 87–110.

[10] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. 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, Springer, 2016, pp. 327–357.

[11] G. Botrel, T. Piellard, Y. E. Housni, A. Tabaie, G. Gutoski, and I. Kubjas, *Consensys/gnark-crypto: v0.11.2*, Jan. 2023.

[12] S. Bowe, J. Grigg, and D. Hopwood, *Recursive proof composition without a trusted setup*, Cryptology ePrint Archive, (2019).

[13] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, *Bulletproofs: Short proofs for confidential transactions and more*, in 2018 IEEE symposium on security and privacy (SP), IEEE, 2018, pp. 315–334.

[14] B. Bünz, M. Maller, P. Mishra, N. Tyagi, and P. 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, Springer, 2021, pp. 65–97.

[15] D. Catalano and D. 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, Springer, 2013, pp. 55–72.

[16] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. 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, Springer, 2020, pp. 738–768.

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

[18] D. Deuber, N. Döttling, B. Magri, G. Malavolta, and S. A. K. 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, Springer, 2020, pp. 315–334.

[19] J. Dreier, P. Lafourcade, and Y. Lakhnech, *Formal verification of e-auction protocols*, in International Conference on Principles of Security and Trust, Springer, 2013, pp. 247–266.

[20] Y. El Housni and G. Botrel, *Edmsm: multi-scalar-multiplication for snarks and faster montgomery multiplication*, Cryptology ePrint Archive, (2022).

[21] D. Escudero, V. Goyal, A. Polychroniadou, and Y. Song, *Turbopack: honest majority mpc with constant online communication*, in Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, 2022, pp. 951–964.

[22] A. Fiat and A. Shamir, *How to prove yourself: Practical solutions to identification and signature problems*, in Conference on the theory and application of cryptographic techniques, Springer, 1986, pp. 186–194.

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

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

[25] ———, *Proposer builder separation (pbs) - ethereum roadmap*. https://ethereum.org/en/roadmap/pbs/, 2024.

[26] G. Fuchsbauer, E. Kiltz, and J. 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, Springer, 2018, pp. 33–62.

[27] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, *Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge*, Cryptology ePrint Archive, (2019).

[28] N. Glaeser, I. A. Seres, M. Zhu, and J. Bonneau, *Cicada: A framework for private non-interactive on-chain auctions and voting*, Cryptology ePrint Archive, (2023).

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

[30] U. Haböck, *Multivariate lookups based on logarithmic derivatives*, Cryptology ePrint Archive, (2022).

[31] F. Hao and P. Zieliński, *A 2-round anonymous veto protocol*, in International Workshop on Security Protocols, Springer, 2006, pp. 202–211.

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

[33] J.-H. Im, T.-Y. Youn, and M.-K. Lee, *Privacy-preserving blind auction protocol using fully homomorphic encryption*, Advanced Science Letters, 22 (2016).

[34] A. Kate, G. M. Zaverucha, and I. 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, Springer, 2010, pp. 177–194.

[35] A. Kavousi, Z. Wang, and P. Jovanovic, *Sok: Public randomness*, in 2024 IEEE 9th European Symposium on Security and Privacy (EuroS&P), IEEE, 2024, pp. 216–234.

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

[37] M. Król, A. Sonnino, A. Tasiopoulos, I. Psaras, and E. Rivière, *Pastrami: privacy-preserving, auditable, scalable & trustworthy auctions for multiple items*, in Proceedings of the 21st International Middleware Conference, 2020, pp. 296–310.

[38] G. Malavolta and S. A. K. Thyagarajan, *Homomorphic time-lock puzzles and applications*, in Annual International Cryptology Conference, Springer, 2019, pp. 620–649.

[39] J. Milionis, D. Hirsch, A. Arditi, and P. Garimidi, *A framework for single-item nft auction mechanism design*, in Proceedings of the 2022 ACM CCS Workshop on Decentralized Finance and Security, 2022, pp. 31–38.

[40] J. A. Montenegro, M. J. Fischer, J. Lopez, and R. Peralta, *Secure sealed-bid online auctions using discreet cryptographic proofs*, Mathematical and Computer Modelling, 57 (2013), pp. 2583–2595.

[41] M. Naor, B. Pinkas, and R. Sumner, *Privacy preserving auctions and mechanism design*, in Proceedings of the 1st ACM Conference on Electronic Commerce, 1999, pp. 129–139.

[42] V. Nikolaenko, S. Ragsdale, J. Bonneau, and D. Boneh, *Powers-of-tau to the people: Decentralizing setup ceremonies*, in International Conference on Applied Cryptography and Network Security, Springer, 2024, pp. 105–134.

[43] A. Nitulescu, *zk-snarks: A gentle introduction*, Ecole Normale Superieure, (2020).

[44] T. P. Pedersen, *Non-interactive and information-theoretic secure verifiable secret sharing*, in Annual international cryptology conference, Springer, 1991, pp. 129–140.

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

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

[47] K. Sako, *An auction protocol which hides bids of losers*, in International workshop on public key cryptography, Springer, 2000, pp. 422–432.

[48] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, *Zerocash: Decentralized anonymous payments from bitcoin*, in 2014 IEEE symposium on security and privacy, IEEE, 2014, pp. 459–474.

[49] J. C. Schlegel, *Transaction ordering auctions*, arXiv preprint arXiv:2312.02055, (2023).

[50] J. Thaler et al., *Proofs, arguments, and zero-knowledge*, Foundations and Trends® in Privacy and Security, 4 (2022), pp. 117–660.

[51] N. Tyagi, A. Arun, C. Freitag, R. Wahby, J. Bonneau, and D. Mazières, *Riggs: Decentralized sealed-bid auctions*, in Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, 2023, pp. 1227–1241.

[52] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, *Scalable anonymous group communication in the anytrust model*, in European Workshop on System Security (EuroSec), vol. 4, 2012.

[53] F. Wu, T. Thiery, S. Leonardos, and C. Ventre, *Strategic bidding wars in on-chain auctions*, arXiv preprint arXiv:2312.14510, (2023).

[54] P. Xia, H. Wang, Z. Yu, X. Liu, X. Luo, and G. Xu, *Ethereum name service: the good, the bad, and the ugly*, arXiv preprint arXiv:2104.05185, (2021).

[55] J. Xiong and Q. Wang, *Anonymous auction protocol based on time-released encryption atop consortium blockchain*, arXiv preprint arXiv:1903.03285, (2019).

[56] S. Yang, K. Nayak, and F. Zhang, *Decentralization of ethereum's builder market*, arXiv preprint arXiv:2405.01329, (2024).

[57] S. Yuan, J. Wang, B. Chen, P. Mason, and S. 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, 2014, pp. 1897–1906.

[58] A. Zapico, V. Buterin, D. Khovratovich, M. Maller, A. Nitulescu, and M. Simkin, *Caulk: Lookup arguments in sublinear time*, in Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, 2022, pp. 3121–3134.

[59] H. Zhang, M. Yeo, V. Estrada-Galinanes, and B. Ford, *Zeroauction: Zero-deposit sealed-bid auction via delayed execution*, Cryptology ePrint Archive, (2024).

[60] K. Zhong, Y. Ma, Y. Mao, and S. Angel, *Addax: A fast, private, and accountable ad exchange infrastructure*, in 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI'23), 2023, pp. 825–848.

[61] R. Zippel, *Probabilistic algorithms for sparse polynomials*, in International symposium on symbolic and algebraic manipulation, Springer, 1979, pp. 216–226.

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

Appendix

A. Auction Definitions and Proofs

An auction protocol takes as input bids $b_1, \ldots, b_m$ in a domain χ and outputs the highest bid $b_w = \max\{b_1, \ldots, b_m\}$ as winner. Inspired by [28], we now provide a formal definition for a (verifiable and private) sealed-bid auction Π_{Auction}.

Definition 2 (Selead-bid auction)**.** A sealed bid auction $\Pi_{\mathsf{Auction}} = (\mathsf{Setup}, \mathsf{Seal}, \mathsf{Eval}, \mathsf{Verify})$ is defined with the following algorithms:

- $(\mathsf{SP}) \leftarrow \mathsf{Auction.Setup}(1^\lambda)$: Takes as input a security parameter λ, and outputs system parameters SP.
- $(c_i, \pi_i) \leftarrow \mathsf{Auction.Seal}(\mathsf{SP}, i, b)$: 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.
- $(y, \pi) \leftarrow \mathsf{Auction.Eval}(\mathsf{SP}, \{c_i, \pi_i\}_{i\in[m]})$: 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.
- $(1/0) \leftarrow \mathsf{Auction.Verify}(\mathsf{SP}, y, \pi_w)$: Verifies if the output y is indeed the correct result of the auction b_w.

Definition 3 (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 $b_1, \ldots, b_m \in \chi$, we have:

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

Definition 4 (Soundness)**.** Let b_w denote the highest (honest) bid and assume the adversary $\mathcal{A}$ corrupts the auctioneer and

Input: h bids $b_1, \ldots, b_h$ from honest bidders and k bids $b'_1, \ldots, b'_k$ from malicious bidders (the exact order is irrelevant).
Output: (b^*, sp), where

- Compute $b^* = \max(b_1, \ldots, b_h, b'_1, \ldots, b'_k)$.
- 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.

Fig. 8: Ideal Functionality $\mathcal{F}$ for Cryptobazaar

$k \leq m - 2$ 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 $\lambda \in \mathbb{N}$, we have:

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

Definition 5 (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 s.t.:

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

1) Proof for Theorem 1: *Completeness.* Follows directly from the underlying AV protocol. The value $Z = (x + b \circ r) \circ Y$ sent by each bidder in the bidding phase, see Figure 2, is the same as the one sent in the original AV, where here the randomness r comes into 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 describing our formal privacy analysis we argue about the privacy of the underlying AV protocol. When x_i is sampled randomly by an honest bidder i, the adversary controlling all but one other bidder (*i.e.,* that is $k \leq m - 2$) cannot break the privacy of the bidder i (Theorem 4, [31]) and the confidentiality of its input. This is because the value Y_i has a uniform distribution (Lemma 3, [31]) and under the DDH assumption one cannot distinguish between $x_i Y_i$ and a random group element $r_i Y_i$. 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 [29]. 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 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 bid, and second highest (honest) bid. The description of ideal functionality $\mathcal{F}$ and simulator $\mathcal{S}$ are given in Figure 8 and Figure 9, respectively. We define a sequence of hybrid distributions starting from the actual protocol transcript and ending with the simulated transcript. 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.

$\underline{\mathsf{Hybrid}_0}$ This is the actual view of Cryptobazaar.

$\underline{\mathsf{Hybrid}_1}$ 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 $(\tilde{\mathsf{x}}, \tilde{\mathsf{q}}, \tilde{\mathsf{b}}, \tilde{\boldsymbol{X}}, \tilde{\pi}_x, \tilde{\pi}_r, \tilde{\pi}_b)$ 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 $\tilde{\mathsf{b}}$. Note that the original KZG commitment is deterministic in the sense that the same polynomials have the same commitments. So, for the indistinguishably to hold we need to use blinded KZG.

$\underline{\mathsf{Hybrid}_2}$ 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 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 distribution (Lemma 3, [31]).

$\underline{\mathsf{Hybrid}_3}$ 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 $\tilde{\pi}_{Z_i}$.

$\underline{\mathsf{Hybrid}_4}$ The same as Hybrid_3, except that the output vector

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

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. 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 $\mathcal{A}$.
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 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.

Fig. 9: Simulator for Cryptobazaar

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 $\tilde{\pi}_{b^*}$ is enough to ensure the privacy of honest bidders and failure to do so by $\mathcal{A}$ does not affect the security of the view. Thus, this view is indistinguishable from the previous one.

$\underline{\textsf{Hybrid}_5}$ The same as $\textsf{Hybrid}_4$, 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 due to the zero-knowledge property of $\tilde{\pi}_w$.

$\underline{\textsf{Hybrid}_6}$ This is the view of $\mathcal{A}$ simulated by $\mathcal{S}$. 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.

2) Proof for Lemma 4: Completeness is immediately followed by writing out the verification equation as $e(G, H)^{f(\tau) - f(\gamma) + \gamma q(\gamma)} = e(G, H)^{\tau q(\tau)}$.

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 $\mathcal{S}_{\textsf{pv}}$ 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. 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.

3) Proof for Lemma 5: We show knowledge soundness by building an efficient extractor $\mathcal{E}$ that extracts the witness and zero-knowledge by building 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 $\mathtt{t} = (\mathtt{c} - [y]_1)(\tau - \gamma)$.

4) Proof for Lemma 6: 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 $\boldsymbol{b}_j = \boldsymbol{b}_n = 1$.

5) Proof for Lemma 7: We show how to build an efficient extractor $\mathcal{E}$ 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, $\mathcal{A}$ sends polynomial $f(X)$. Then, $\mathcal{E}$ computes $u(w)$ from $f(1)$ and $u(1)$ and all $u(w^i)$ from $f(w^{i-1})$ and $u(w^{i-1})$.

Zero knowledge. We construct $\mathcal{S}_{\textsf{unary}}$ which, given instance $\mathtt{u}$, toxic waste τ and the verifier's challenge, produces a transcript that is identically distributed to the transcript obtained from the interaction with an honest prover that has a witness. Polynomials $u(X), f(X), B(X)$ are 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.

6) Proof for Lemma 8: Given that our protocol is closely similar to the generalized IPA introduced in Bunz et al. [14], we just give a high level intuition and refer the reader to [14] 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 [14]. 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 $i \in (\log n, \log n - 1, \dots, 0)$.

7) Proof for Lemma 9: 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. We define a game involving an algebraic adversary $\mathcal{A}$ and an efficient extractor $\mathcal{E}$. Given SP, $\mathcal{A}$ 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 $\mathtt{s}_1, \mathtt{s}_2, \mathtt{B}, \mathtt{R}, \mathtt{q}, \mathtt{D}, \mathtt{Q}_\mathtt{s}$ 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 KZG checks implies that by the knowledge soundness of KZG and Schwartz-Zippel all polynomial identities hold except with negligible probability. Since the 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 $\sum_{h \in \mathbb{H}} s_i(h) = 0$. Thus, $\sum_{h \in H} B(h) + ca(h)b(h) = z_1$. Now suppose that $\sum_{h \in \mathbb{H}} a(h) \cdot b(h) = x' \neq x$ and 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 $c = \frac{p'-p}{x-x'}$ happening 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 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 enables $\mathcal{E}$ to interpolate $a(X), b(X)$. From witness extended emulation [50] we know that $\mathcal{E}$ is efficient and fails only with negligible probability.

Zero knowledge. 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:

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) $\mathtt{Q}_\mathtt{s}$ 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.

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

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 $\mathtt{Q}_\mathtt{s} = [a_6]_1$.
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 $q_T = q_6 + vq_7 + v^2q_8$.

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

B. Artifact Evaluation

In this section, we provide all of the necessary information to set up and evaluate the artifact associated with the paper *Cryptobazaar: Private Sealed-bid Auctions at Scale* and verify the presented results.

1) Description & Requirements:

a) How to Access: The Cryptobazaar artifact can be downloaded from the following locations:

- Zenodo: https://doi.org/10.5281/zenodo.17817520
- GitHub: https://github.com/akinovak/cryptobazaar-impl

We recommend using the code archived on Zenodo to ensure that you are using the correct version. If you prefer using GitHub, please make sure to checkout GitHub Release v1.0.0 which corresponds to the artifact on Zenodo.

b) Hardware Dependencies: None.

c) Software Dependencies: Mandatory: Rust. Optional: Git and Docker.

d) Benchmarks: None.

2) Artifact Installation & Configuration:

To prepare your system for evaluating the Cryptobazaar artifact, consider the following steps.

a) Install Rust: Cryptobazaar is written in the programming language Rust. To install Rust, please refer to the following website: https://rust-lang.org/tools/install/. Rust will download and install all of the additional artifact dependencies automatically during the artifact build process.

b) Install Git (Optional): To retrieve the Cryptobazaar source code from GitHub, we recommend using Git. In case Git is not already installed on your system, please refer to your system's package manager or to the following website: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git.

c) Install Docker (Optional): The Cryptobazaar artifact benchmarks can also be executed via the container platform Docker. In case Docker is not already installed on your system, please refer to your system's package manager or to the following website: https://docs.docker.com/get-started/.

d) Download Repository: Download the repository containing the Cryptobazaar artifact from Zenodo or GitHub. To clone the repository with Git and checkout the correct version, execute:

```
git clone https://github.com/akinovak/cryptobazaar-impl
cd cryptobazaar-impl
git checkout -b cryptobazaar v1.0.0
```

e) Build Artifact (Optional): To build the Cryptobazaar artifact and all of its dependencies, execute:

```
cd cryptobazaar-impl
cargo build --release
```

This step is optional since the benchmarking process described in the next section will automatically build the artifact and download and install all of the dependencies.

3) Major Claims:

In the paper, we claim the following contributions:

C1 We propose Cryptobazaar, a private scalable sealed-bid auction protocol that supports first- and second-price auctions and relies only on an untrusted auctioneer for coordination.

C2 We co-design various novel (public coin) zero-knowledge succinct arguments of knowledge to ensure soundness and privacy of the different protocol steps.

C3 We present two extensions of Cryptobazaar showing how to run uniform auctions and sequential/iterative auctions efficiently without having to re-execute the full protocol.

C4 We demonstrate the practicality of our system by evaluating an open-source implementation of Cryptobazaar in Rust. We show that Cryptobazaar scales to hundreds of bidders and large price ranges while remaining efficient enough for a practical deployment.

While claims C1-C3 are supported by the design details and security proofs presented in the paper, we aim to substantiate C4 through the artifact evaluation.

TABLE III: 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

4) Evaluation:

There are six different microbenchmarks that we report in the Cryptobazaar evaluation section. These benchmarks were generated on an Apple MacBook Pro with an M1 Max chip with 10 cores and 64 GB memory. For the sake of being self-contained, we re-report the evaluation results in Table III. Note that the benchmarks are platform-dependent and thus the total running time may vary on different systems. For example, Intel platforms can utilize AVX instructions (enabled by default via the `asm` feature in the `Cargo.toml` file) to speed up certain cryptographic operations.

a) Experiment: [1 human minute + 60 compute minute] This experiment computes all of the microbenchmarks measuring the computational overheads (in ms) of the four validity proofs, the AV protocol, and the creation of the auction results vector.

[Preparation] None.

[Execution] From the Cryptobazaar directory, execute

```
./run-benchmarks.sh
```

Alternatively, the benchmarks can be run via Docker by executing `docker build .` in the repository root folder.

[Results] As presented in Table III. To provide a brief interpretation of the results: For example, for 128 bidders and a price range of 8192, Cryptobazaar incurs a computational overhead of $1880\,\mathrm{ms}$ per bidder per auction run and the protocol terminates in $431\,\mathrm{ms}$. All of the benchmarked configurations of Cryptobazaar are practical enough to satisfy the requirements of real-world systems. A run of the Cryptobazaar auction protocol with 128 bidders and a price range of 1024 completes in less than 0.5 seconds which, for example, satisfies the requirement of terminating within Ethereum's block time window of 12 seconds.