Riggs: Decentralized Sealed-Bid Auctions

Nirvan Tyagi
Cornell University

Arasu Arun
New York University

Cody Freitag
Cornell Tech

Riad Wahby
Carnegie Mellon University
Cubist

Joseph Bonneau
New York University
a16z crypto research

David Mazières
Stanford University

ABSTRACT

We introduce the first practical protocols for fully decentralized sealed-bid auctions using timed commitments. Timed commitments ensure that the auction is finalized fairly even if all participants drop out after posting bids or if $n-1$ bidders collude to try to learn the n^{th} bidder's bid value. Our protocols rely on a novel non-malleable timed commitment scheme which efficiently supports range proofs to establish that bidders have sufficient funds to cover a hidden bid value. This allows us to penalize users who abandon bids for *exactly* the bid value, while supporting simultaneous bidding in multiple auctions with a shared collateral pool. Our protocols are concretely efficient and we have implemented them in an Ethereum-compatible smart contract which automatically enforces payment and delivery of an auctioned digital asset.

1 INTRODUCTION

Sealed-bid auctions are an important tool in auction design. The most well-known format is the *sealed-bid second-price auction*, also known as a *Vickrey auction* in honor of Vickrey's Nobel prize–winning 1961 work formalizing the concept [Vic61] (though such auctions were regularly used in practice in the 19th century [LR00], well before Vickrey's work). In a classic (offline) Vickrey auction, a group of n bidders privately submits bids to a trusted auctioneer, who awards the good to the highest bidder at the price of the second-highest bid submitted. Vickrey showed that, in an idealized model, bidders in this auction format are incentivized to bid their true valuation of the good and the resulting sale price is equivalent to that produced by the more commonly used open-bid ascending-price auction (or *English auction*).

Vickrey auctions only require one round of communication, a compelling efficiency advantage over English auctions which can require an extended bidding period to discover the winning price. Vickrey auctions also have a privacy advantage: only the auctioneer need see bid values. English auctions inherently require publicizing bids to enable price discovery.

Despite these advantages, Vickrey auctions have remained far less common in practice. Rothkopf et al. [RTK90] argued in 1990 that two key issues prevented more widespread use: concerns about cheating by the trusted auctioneer and concerns by bidders about revealing their true valuations. For example, just this year, Google is being sued by the US Justice department for "anticompetitive auction manipulation" [Goo23].

In 1993, Nurmi and Salomaa [NS93] first proposed using cryptography to prevent cheating by the auctioneer and limit public revelation of bids. Franklin and Reiter [FR96] proposed the first complete cryptographic protocol for sealed-bid auctions, relying on an honest majority of auctioneers to ensure honest behavior. Sealed-bid auctions have since motivated a diverse cryptographic literature with dozens of proposed protocols, covering many different auction formats and privacy models (for a survey see Alvarez and Nojoumian [AN20]). Generically, the auction process can be cast as a secure multi-party computation problem between the bidders [NPS99]; MPC was famously deployed in a real 2009 auction for Danish sugar beets [BCD+09].

Prior to the advent of blockchains, all of these protocols suffered from a fundamental limitation that the cryptographic protocol could only compute the correct sale price but not enforce that the winning bidder actually pays (nor that the item is delivered as promised). Blockchains with a sufficiently powerful smart contract environment can facilitate a fully *decentralized auction* by enforcing payment and even enforcing delivery for certain types of digital goods (e.g., NFTs). Unsurprisingly, cryptographic auctions were quickly suggested as an example application for smart-contract enabled blockchains [KMS+16, BK18].

The typical structure sees users post cryptographic commitments to the blockchain to bind them to a (hidden) bid value. Users then reveal their bids after all commitments have been published. However, blockchains cannot force any user to reveal their committed bid. Bidders can be incentivized to open their bid via penalties [ADMM14, KZZ16] or required to provide enough information for a majority of other bidders to reconstruct their bid [BK18]. But these approaches are complex in practice, requiring either careful reasoning about the size of penalties required to incentivize correct behavior or opening the auction up to attack by a dishonest majority of participants.

In this work, we refine the approach of implementing sealed-bid auctions using *timed commitments*, as first proposed by Boneh and Naor [BN00]. With timed commitments, a participant's bid can be recovered without the participant's cooperation, by computing a slow function which *forces open* their commitment. This approach guarantees that the auction can be fairly concluded even if all parties drop out or if $n-1$ bidders collude. The slowness of the force-opening algorithm is necessary to ensure bidders cannot learn others' bids in time to adjust their own bidding strategy.

The basic proposal of Boneh and Naor does not achieve fully decentralized auctions, however, as it does not defend against bidders placing bids they cannot afford. This problem is more than merely a nuisance: a malicious bidder can manipulate their balance during the bid-opening phase to back out of a bid by nullifying their ability to pay. Such manipulation has implications on the *credibility* of an auction [AL20] in which a set of bidders may collude with the seller to bias auction results. In this work, we extend timed commitment–based techniques to the fully decentralized sealed-bid setting. Specifically, we develop new techniques for efficient proofs that a bid sealed via a timed commitment is fully backed by a bidder's available funds, while keeping the actual bid amount private. Our proposed solutions (which we collectively call Riggs) support concurrent bids on multiple asynchronous auctions and are extensible to any auction design in the sealed bid setting. This includes not just second-price Vickrey auctions, but also, for example, generalized $k+1^{\text{st}}$ price multi-good auctions or multi-round simultaneous ascending auctions as used by the FCC for wireless spectrum [GV99].

We implement our designs in Rust and build a compatible smart contract implementation in Solidity that adheres to existing token standards and can be deployed on the Ethereum blockchain to interact with the NFT and token ecosystems. The cost to generate a bid in our Rust implementation is 71 ms; the cost to validate that bid in our Ethereum smart contract implementation is roughly 2.5 million gas. While the latter is expensive (≈\$134 on Ethereum at time of writing), we believe that these costs will decrease substantially in the future; we discuss in Section 7.

To summarize, our contributions are:

- Introducing the threat model for a fully decentralized auction and uncovering weaknesses in folklore proposals for the setting.
- Construction of a new non-malleable timed commitment scheme that efficiently supports range proofs. With practicality in mind, we design and prove our scheme secure in the random oracle model. Thus, even when used without range proofs, our timed commitment scheme is the most efficient that we are aware of.
- Design of the Riggs-RP and Riggs-TC variants of an auction protocol for the decentralized setting.
- Implementation and evaluation of our protocols in a native Rust implementation and a compatible smart contract implementation deployable on the Ethereum blockchain.

2 BACKGROUND AND PRELIMINARIES

Prime-order cyclic groups. We denote $\mathbb{G}$ as a *cyclic group of prime order*. Looking forward, the group will be selected to have order p determined by the range of valid bids and a security parameter λ. We denote canonical generators of the group as g and h, and we assume an efficient setup algorithm that on input security parameter λ, generates a group, $(p, \mathbb{G}, g, h) \leftarrow\!\!\$\ \mathsf{GGen}(\lambda)$, where $\|p\| = \lambda$. The discrete log of g is not known with respect to h.

Pedersen commitments. A *Pedersen commitment* [Ped91] commits to a message $m \in \mathbb{Z}_p$ in a commitment $com = g^m h^\alpha$ in a prime-order cyclic group for a random $\alpha \leftarrow\!\!\$\ \mathbb{Z}_p$. The commitment com can be opened to reveal m by providing *opening proof* α and m. The commitment is *hiding* and *binding* meaning no information about m is revealed through com and it is infeasible for computationally bounded adversaries to open com to any value other than m. Pedersen commitments have a convenient *additive homomorphic* property that we will take advantage of. That is, two commitments, $com_1 = g^{m_1} h^{\alpha_1}$ and $com_2 = g^{m_2} h^{\alpha_2}$ can be combined to compute $com \leftarrow com_1 \cdot com_2$ that opens to $m_1 + m_2$ using opening $\alpha_1 + \alpha_2$.

RSA groups. A *strong RSA group* is the multiplicative group of invertible integers modulo N (denoted $\mathbb{Z}_N^* = \{x \in \mathbb{Z}_N : \gcd(x, N) = 1\}$), where N is the product of two safe primes q_1, q_2 (i.e., such that $\frac{q_1-1}{2}$ and $\frac{q_2-1}{2}$ are also prime). We define the group of *RSA quadratic residues* for N as $\mathbb{QR}_N = \{x^2 \mod N : x \in \mathbb{Z}_N^*\}$, and the *signed quadratic residues* as $\mathbb{QR}_N^+ = \{|x| : x \in \mathbb{QR}_N\}$. In $\mathbb{QR}_N^+$, the elements $N - x$ and x are equivalent, so the group can be represented by integers in the interval $[1, \lfloor \frac{N}{2} \rfloor]$. Furthermore, all elements can be efficiently tested for membership and all elements (except for 1) are part of subgroups of size $> \frac{\varphi(N)}{4}$, where $\phi(\cdot)$ is the Euler totient function. For this reason, we will use the group $\mathbb{G} = \mathbb{QR}_N^+ \setminus \{1\}$, for which we assume a setup algorithm, $(q_1, q_2, N, \mathbb{G}, g, h) \leftarrow\!\!\$\ \mathsf{RSAGGen}(\lambda)$ where $\|q_1\| = \|q_2\| = \tau(\lambda)$; and $\tau(\lambda)$ is defined such that factoring N takes time 2^λ. We refer to [Pie19, MT19] for more details.

Timed commitments. A *timed commitment* [BN00] (or similar *time-lock puzzle* [RSW96]) commits to a message m in a commitment com such that the message is hidden, but can be force-opened by any party by performing a sequential amount of work determined by some delay parameter t. Timed commitments can be created and opened efficiently by the committer (requiring at most $O(\lg t)$ work). Furthermore, if a commitment is force-opened, an efficiently-verifiable proof of opening can be provided.

Most timed-commitment protocols rely on the decisional repeated squaring problem [RSW96]: given $g \in \mathbb{G}$, it is hard to distinguish $z = g^{2^t}$ from random without executing a computation of sequential depth at least t. This problem is believed to be as hard as computing the order of the group (for an RSA group, factoring N); this has been proven in generic computation models [KLX20, RS20].

Boneh and Naor's original construction [BN00] required the committer (but not others) to know the group order $\|\mathbb{G}\|$ and hence required a different $\mathbb{G}$ for each committer. Modern timed commitments [FKPS21, KLX20, MT19, TCLM21] use of proofs of exponentiation in groups of unknown order [Wes19, Pie19], in which a prover convinces a verifier for $z, g \in \mathbb{G}$ and $\alpha \in \mathbb{Z}$, the relation $z = g^\alpha$ holds. Importantly, the integer α can be much larger than $\|\mathbb{G}\|$, but the verifier's running time remains $\tilde{O}(\log\|\mathbb{G}\|)$. Thus these modern constructions enable all committers to use the same $\mathbb{G}$ which can be a global parameter.

Non-interactive zero-knowledge proofs. A *non-interactive proof system* for a relation $\mathcal{R}$ over *statement-witness* pairs (x, w) enables producing a proof, $\pi \leftarrow \mathsf{Prove}(pk, x, w)$, that convinces a verifier $\exists w : (x; w) \in \mathcal{R}$, $0/1 \leftarrow \mathsf{Ver}(vk, \pi, x)$; pk and vk are proving and verification keys output by a setup, $(pk, vk) \leftarrow \mathsf{Keygen}(\mathcal{R})$. A *non-interactive argument of knowledge* further convinces the verifier not only that the witness w exists but also that the prover *knows* w (also known as *soundness*). If proved in *zero-knowledge*, the verifier does not learn any additional information about w.

Range proofs. We make use of non-interactive zero-knowledge *range proofs* which allow a prover to convince a verifier that a committed integer x falls within a range $[A, B]$. There have been two standard approaches to constructing range proofs. The first is based on n-ary decomposition of x (or $x - A$ and $B - x$), committing to decomposed limbs, using homomorphic properties of the commitment to show the limbs recompose to x, and lastly proving that each limb falls within $[0, n]$ [CCS08, Gro11, BBB$^+$18, CHJ$^+$20]. The second is based on square decomposition of x using Lagrange's four square theorem which states that every positive integer can be decomposed to the sum of four integer squares, $x = \sum_{i=1}^{4} x_i^2$. The prover homomorphically computes a commitment to $x - A$ and $B - x$ and proves that both are positive by additionally providing commitments to the square decomposition and homomorphically verifying the sum of squares [Bou00, Lip03, Gro05, CPP17]. This approach requires commitments to integers which has typically necessitated using hidden order groups [FO97, DF02], until recently Couteau et al. constructed bounded integer commitments in known order groups [CKLR21]. Our techniques for constructing range proofs on top of timed commitments will be compatible with either approach.

3 OVERVIEW

3.1 Auction Setting and Threat Model

In this work, we are concerned with building *decentralized sealed-bid auctions*. In place of a trusted auctioneer used in traditional sealed-bid auctions, we rely on a decentralized consensus protocol (in short, a blockchain). This could be done using a special-purpose consensus protocol implementing the auctioneer logic; though our goal is to use a general-purpose consensus protocol that supports arbitrary programs (or *smart contracts*).

We do not rely on any specific underlying consensus mechanism, e.g., proof-of-work or proof-of-stake. We assume, as is standard, that the underlying consensus protocol is *correct*, that is, only valid transactions which follow the rules of the smart contract can be added to the chain. We also assume that the consensus protocol is *eventually consistent*, that is, all nodes agree on transaction history except up to a small suffix. Finally, we assume that the underlying consensus protocol is *live* and *censorship-resistant*. That is, users attempting to broadcast a transaction to the chain will succeed with high probability, possibly under the assumption that adequate fees are paid. No auction system will be secure if an attacker can manipulate the consensus protocol (for example, via bribery of the participants) to prevent anybody else from placing a bid.

We model bidders and sellers as pseudonymous, possibly ephemeral cryptographic identities. An adversary may control any number of Sybil identities, but cannot impersonate an identity they do not own. Since identities are pseudonymous, the auction mechanism cannot rely on reputational or legal pressure to induce exchange between the winner(s) and seller; enforcement must therefore be integrated into the protocol.

Our system can tolerate targeted network-level denial-of-service attacks, e.g. arbitrarily dropping or delaying packets. As long as we assume that a new ephemeral identity is able to send at least one transaction (a bid commitment) to the blockchain before being targeted for denial of service, our use of timed commitments ensures an attacker cannot manipulate the auction result by preventing them from publishing their bid opening.

3.2 Technical Overview

Our main technical contribution is a cryptographic protocol that composes timed commitments with efficient range proofs. Using this protocol, we avoid pitfalls in previously proposed auction systems stemming from abandoned bids, high collateral cost, and denial-of-service attacks. We provide an overview of our approach by stepping through a series of strawman solutions illustrating each of these issues.

"Commit-and-reveal" with a per-auction collateral. As a starting point, consider a natural two-phase construction in which bidders first submit a commitment to their bid in a *bid collection* phase, and then open the commitment to reveal their bid to conclude the auction in a later *bid self-opening* phase. A serious problem with this approach is that bidders must be trusted to open their commitment: a cheating bidder can watch the bids as they are opened during the second phase and simply refuse to open theirs if they do not like the outcome, thus biasing the auction. Consider the following concrete attack against second-price (Vickrey) auctions. Cheating bidders in collusion with the seller (or Sybils controlled by the seller) can place multiple bids at various price points. As honest bids are revealed, the cheating bidder can reveal a bid that is just below the highest honest bid and abandon other bids, thus driving up the price paid to the seller. This is a standard attack in the economics literature against the credibility of an auction [AL20]. In the worst case this attack reduces the auction to a first-price auction which is a poor outcome – it incentivizes bidders not to bid their actual value, hindering efficient price discovery.

To incentivize against such abandoned bids, we might require bidders to place collateral in escrow (via a smart contract) before bidding, which is forfeit if the bidder does not self-open. In this case, the collateral must be larger than the corresponding bid so that the seller can collect payment; an opened bid greater than its collateral is invalid. It has been shown that setting collateral in this manner recovers credibility of an auction [FW20].

However, even this collateral strawman proposal has two limitations that motivate our work. First, unless the escrow contract has some privacy mechanism, collateral values are public—so to avoid revealing information, the user must escrow much more than the bid amount (say, an upper bound on the item's value). Schlegel et al. [SM21] analyzed this setup from an economic point of view and concluded that it has the potential to skew bidding strategy. It is also inefficient, as users must lock up a large amount of extra collateral to avoid revealing information. This is even more severe with simultaneous auctions, each of which require separate collateral per bid. In total, several concurrent bids may require the user to escrow a huge amount, potentially limiting their ability to participate.

Second, while collateral incentivizes a bidder not to abandon their bid, it does not address scenarios where a bidder tries to open their bid but is unable to. As examples, a user may have unreliable network access, or an auction may be held on a blockchain that experiences transaction congestion. In these and similar situations, the risk of losing collateral may dissuade users from participating. Perhaps more importantly, bidders have an incentive to mount denial-of-service attacks against one another: an attacker can identify other bidders from their posted bid commitment and interfere with future self-opening posts, thereby reducing the pool of bids in the auction. We now address these limitations in turn.

Pooled collateral and concurrent auctions. Using a blockchain platform that doesn't support transaction-level privacy inherently requires that users have more collateral on deposit than the maximum plausible price for an item. However, if a user is bidding on multiple items in concurrent auctions, the user can escrow a single collateral pool for all of their active bids, potentially greatly reducing their escrow requirements.

We could try pooling collateral with a simple rule that a user's bid is invalid if it exceeds the collateral size, otherwise the bid value (once revealed) is subtracted from the bidder's collateral pool. Unfortunately, this enables bidders to abandon their bids: during the opening phase, a remorseful bidder can race to win a separate item in a wash-sale auction (i.e., creating a short-term auction in which they are the seller), thereby siphoning enough collateral from their pool to invalidate the regretted bid.

This attack relies on the fact that concurrent auctions' start and end times can be different, allowing a wash-sale auction to complete during another auction's opening phase. Thus, we might prevent this attack by forcing auctions to be synchronized in epochs: all auctions within an epoch start and end together, and the auctions for the next epoch cannot start until the prior epoch is finished. In this case, a user's bids are either all valid (i.e., bids sum to less than the collateral pool) or all invalid, and a user's collateral pool is locked during any epoch in which the user placed a bid. We give further details on this approach in Appendix B.

The requirement to synchronize the start and end of all auctions is inconvenient, both for bidders and sellers. To sidestep this requirement, an alternative approach is to ask bidders to *prove* in zero knowledge that the sum of their active bids is less than their collateral (reminiscent of private payment systems [BAZB20]). In this model, a bidder cannot place any bid that would exceed the size of their collateral pool (and hence invalidate a prior bid), since in that case they would be unable to produce a valid proof—meaning it is safe to allow asynchronous concurrent auctions. Bidders can even adjust the size of their collateral pool as long as they prove that the new amount covers all outstanding bids.

Denial-of-service protection via timed commitments. Boneh and Naor first proposed using timed commitments (§2) for sealed-bid auctions [BN00]. In this arrangement, any abandoned bids that were not opened during the self-opening phase are instead opened during a subsequent *force-opening* phase, via a long sequential computation; the cost to force a commitment must be chosen so that the bid remains secret until all bids are collected. This approach ensures that all bids are included in the auction results, even ones a bidder is unwilling or unable to open. Parties who do the expensive computational work of forcing open abandoned bids can be incentivized with a reward; this reward must be chosen carefully to avoid incentivizing undesirable behavior (e.g., launching denial-of-service attacks purely for the opportunity to collect the forced-opening reward). We now show how to integrate timed commitments with range proofs, which (as discussed above) enable asynchronous concurrent auctions.

Range proofs over timed commitments. A natural approach to integrating timed commitments and range proofs is to use a non-interactive zero knowledge proof system to prove that the enclosed bid lies in the valid range. Many general-purpose zero-knowledge succinct non-interactive argument of knowledge (zkSNARK) constructions are known (e.g., [PHGR13, Gro16, BBHR19, GWC19, CHM+20, Set20]; [Tha20] surveys). In principle, such systems allow one to prove arbitrary relations, and in fact Katz et al. recently used zkSNARKs to ensure non-malleability of timed commitments [KLX20]. In practice, however, the relation being proved must be encoded in an arithmetic constraint formalism, which often introduces orders-of-magnitude overheads for natural computations. We evaluate this approach in Section 7, finding that it results in over 20 second proving times; looking ahead, this is $\approx 300\times$ worse than our solution.

To avoid the overhead of general-purpose proof systems, we devise an application-specific proof system that takes advantage of the algebraic structure of the timed commitment. Our starting point is the linearly-homomorphic timed commitment of [MT19], which is related to Paillier encryption [Pai99]. Given an input-output pair for the sequentially hard repeated squaring problem (h, z) where $z = h^{2^t} \pmod N$, a commitment to bid b is as follows:

$$com(b; \alpha) = \left(h^{\alpha} \pmod N, (1+N)^b \cdot z^{N\cdot\alpha} \pmod {N^2}\right),$$

where α is sampled as a large random exponent. To force open the commitment, one first computes the blinding factor $z^{N\cdot\alpha}$ via repeated squaring of $h^{N\cdot\alpha}$, then unblinds the second element of com and computes the discrete log of the result to the base $(1+N)$. [TCLM21] show how to use a proof of group homomorphism over hidden-order groups [CCL+20, BBF19, BCM05] to prove knowledge of the enclosed bid, ensuring non-malleability.

This construction can be extended to support range proofs. At a high level, the strategy is to take advantage of concretely efficient range proof constructions for Pedersen commitments in a prime-order cyclic group, i.e., $com_{\mathbb{G}} = g^b h^{\beta}$ for random β [BBB+18, CKLR21]. Specifically, rather than simply proving knowledge of the bid enclosed in the timed commitment (as discussed above), one can use a similar proof of group homomorphism to prove that the values committed in com and $com_{\mathbb{G}}$ are equal [WBJP20], then apply a efficient range proof to $com_{\mathbb{G}}$ to show that the timed commitment com contains a bid in the appropriate range.[1] While this construction is much more efficient than using a general-purpose zkSNARK, we can optimize even further in the auction setting.

Timed commitment to range proof commitment opening. The key observation behind our final optimization is that, in our setting, the commitment consistency check can be deferred to the opening phase. Let the timed commitment com's value be the *opening* of the range-proof commitment $com_{\mathbb{G}}$. As before, the range proof must be checked at bid time; later, upon opening the timed commitment, one can simply check that the claimed opening of the range-proof commitment is valid; otherwise, the bid is regarded as invalid. This approach—a timed commitment to the opening of another commitment—was previously proposed for decentralized

[1]Another approach would be to perform the range proof in the RSA group that the timed commitment is in. Unfortunately, range proofs over RSA groups [Bou00, Lip03, Gro05, CPP17] are concretely expensive.

auctions [DDM+20]; we observe that the same mechanism yields very efficient range proofs in our context.

In slightly more detail, the range-proof commitment is a Pedersen commitment $com_{\mathbb{G}} = g^b h^\beta$ to bid b, and the timed commitment's value is (b, β), i.e., the opening of $com_{\mathbb{G}}$. During the bidding phase, any efficient range proof can be used with $com_{\mathbb{G}}$; once the timed commitment is opened, the bid is accepted if and only if its value opens $com_{\mathbb{G}}$.

Importantly, if the timed commitment is non-malleable and binding, the bid's validity is determined at the outset and cannot be changed later. Moreover, all collateral lock-ups are determined by the range-proof commitment, so a valid proof ensures sufficient collateral. The attacker might still submit a malformed timed commitment (i.e., one that does not open the range-proof commitment), but this is not an issue: the effect is just that the attacker's collateral is locked until the bid is invalidated at opening time.

This protocol has significant efficiency benefits over generating range proofs directly on the timed commitment. It obviates the proof of equivalence, thereby allowing the use of non-algebraic timed commitments [FKPS21], which are more efficient. It is also modular: if denial-of-service (DoS) protection is not needed, the timed commitment can just be elided. Putting it all together, we are left with a timed commitment range proof with essentially minimal overhead: it consists solely of the most efficient known non-malleable timed commitment and the most efficient known range proof.

4 NON-MALLEABLE TIMED COMMITMENTS

In this section, we formalize and construct the non-malleable (non-interactive) timed commitments used for our decentralized auction protocols. Our construction builds off of plain commitments by adding a timed trapdoor following [BN00, FKPS21, KLX20]. In Section 4.1, we first define a plain (non-timed) commitment, and then we formalize timed commitments in Section 4.2. Finally, in Section 4.3, we construct a non-malleable timed commitment given any non-malleable plain commitment (e.g. using Pedersen commitments with Bulletproofs [BBB+18]). Full details, discussion, and security proofs for this section are provided in Appendix A.

4.1 Non-interactive Commitments

At a high level, a plain non-interactive commitment consists of a commitment algorithm Comm that on input a bid b outputs a commitment com and an opening proof π_{Open}. Then, a verification algorithm $\mathsf{VerOpen}$ checks whether or not com is a valid commitment to b with respect to π_{Open}. The syntax for a non-interactive commitment C consists of the following algorithms:

- $pp \leftarrow\!\!\$\ \mathsf{C.Setup}(\lambda)$: The setup algorithm defines the public parameters pp given a security parameter λ. We will assume pp is available to all following algorithms, and all parties have assurance it was generated honestly.
- $(com, \pi_{\mathsf{Open}}) \leftarrow\!\!\$\ \mathsf{C.Comm}^{pp}(b)$: The commit algorithm takes in a message b. It produces a commitment com with an opening proof π_{Open}.
- $0/1 \leftarrow \mathsf{C.VerOpen}^{pp}(com, b, \pi_{\mathsf{Open}})$: The opening verification algorithm on input π_{Open} verifies the commitment com opens to the claimed message b. We note that the proof covers the case where the commitment is claimed to be unopenable, i.e., $b = \perp$.

For simplicity, we may drop the public parameters from the superscript if the use is clear from context.

Correctness and security properties. We give high level overviews of the correctness and security properties we require for such commitments. We defer formal definitions to Appendix A.1.

For correctness, we require that for any well-formed pp, $\mathsf{VerOpen}$ outputs 1 on well-formed values output by Comm. We also require that if $\mathsf{VerOpen}$ outputs 1 for some commitment com and bid b, then com is in the support of Comm with bid b. This always can hold by having π_{Open} include the randomness used by Comm and checking that com was computed correctly in $\mathsf{VerOpen}$.

For security, we require two main properties: binding and non-malleability (hiding). All properties hold with high probability over honestly generated public parameters pp. Binding guarantees that no adversary can provide valid opening proofs to open a commitment com to two different values $b \neq b'$.

Non-malleability guarantees that no meddler-in-the-middle (MIM) adversary that receives as input a commitment com for a bid b can output a different commitment com' for a bid b' related to b. In fact, we require a stronger notion of concurrent non-malleability that guarantees the MIM cannot output many different commitments for bids $b_1, \ldots, b_n$ such that they are all jointly related to b in a non-trivial way. We note that non-malleability implies that the commitment satisfies hiding. If an adversary can compute the bid b under a commitment com better than guessing, it could generate a fresh commitment to $b + 1$, for example, which is clearly related to b.

4.2 Timed Commitments

Timed commitments extend plain commitments by adding a "force opening" functionality to open the commitment com after some specified t time, given by the algorithm $\mathsf{ForceOpen}$. We additionally require that "timed" versions of the security properties hold even in the presence of the $\mathsf{ForceOpen}$ algorithm, which we discuss below. A *timed* commitment TC consists of the algorithms $(\mathsf{Setup}, \mathsf{Comm}, \mathsf{ForceOpen}, \mathsf{VerOpen})$, although the syntax of Comm and $\mathsf{VerOpen}$ remain unchanged from plain (non-timed) commitments. The new syntax for Setup and $\mathsf{ForceOpen}$ are as follows:

- $pp \leftarrow\!\!\$\ \mathsf{TC.Setup}(\lambda, t)$: The setup algorithm additionally takes as input a delay parameter t. For definitional simplicity, we assume a fixed delay parameter during setup, but our proposed construction will support flexibly-chosen delay parameters, which we discuss later.
- $(b, \pi_{\mathsf{Open}}) \leftarrow \mathsf{TC.ForceOpen}^{pp}(com)$: The force open algorithm allows any party to recover the message b and an opening proof π_{Open} given the commitment com. The force open algorithm runs in time $t \cdot \mathrm{poly}(\lambda)$. If the commitment fails to open, b is set to $\perp$.

Timed correctness and security properties. Again, we defer the formal definitions to Appendix A.1, but provide a high level overview here. Correctness is the same as for plain commitments, but we also require that on input a well-formed commitment com for bid

b, ForceOpen outputs b. Furthermore, if com is not well-formed, we require that ForceOpen outputs $\perp$.

For security, binding remains unchanged. However, a timed version of non-malleability now must hold even in the presence of the ForceOpen algorithm. Note that the full notion of non-malleability cannot hold in general as ForceOpen immediately breaks hiding (and hence non-malleability) for time t adversaries. As such, we require that MIM attackers running in parallel time less than t cannot maul a commitment com for a bid b into a commitment com' for a related bid b'. Freitag et al. [FKPS21] show that concurrent non-malleability is impossible to achieve in the timed setting. In brief, a MIM attacker that receives as input a commitment com for a bid b can commit to bids $b_1, \ldots, b_n$ that jointly encode com, but this is clearly related to b. So, following the work of [FKPS21], we consider a weaker notion which they term *functional non-malleability* with respect to a class of functions. We consider the class of functions $\mathcal{F}_\ell$ that roughly correspond to functions that can be computed in parallel time less than t and have bounded output length ℓ such that ℓ bits are too short to encode com. We require that for any function $f \in \mathcal{F}_\ell$, no MIM attacker that receives as input a commitment com for a bid b can commit to bids $b_1, \ldots, b_n$ such that $f(b_1, \ldots, b_n)$ is non-trivially related to b. We discuss the subtleties of this definition further in Appendix A.3, but note that (1) this essentially implies the definition of Katz et al. [KLX20] for $\ell = 1$ (formally shown in [FKPS21]) and (2) this suffices for the security of most natural auction types (and all counterexamples for auctions with long output seem to be contrived).

4.3 TTD: Timed Trapdoor Construction

Our main construction is a timed commitment based on RSA groups and proofs of exponentiation and defined relative to an underlying commitment scheme C. Our construction is defined in the random oracle model where all algorithms have access to a random hash function H. Let C be a non-malleable (non-timed) commitment scheme. Looking ahead, we will instantiate C with a Pedersen commitment coupled with Bulletproofs as a proof of knowledge to satisfy non-malleability [BBB⁺18, GT21, GOP⁺22].

On top of the plain commitment scheme, we construct our timed commitment scheme TTD. We set up a “timed trapdoor” based on the sequential hardness of repeated squaring following the seminal work of Rivest, Shamir, and Wagner [RSW96] and additionally using a hash function H (modeled as a random oracle). Specifically, during setup, we initialize an input/ output pair h, z such that $h^{2^t} = z \pmod N$. When computing the commitment, the committer first computes a plain (non-timed) commitment com_{C} for the bid b with opening proof $\pi_{\mathsf{Open},\mathsf{C}}$ using an underlying commitment scheme C. It then sets up a “timed trapdoor” to open com_{C} to the bid b in time t as follows. The committer randomizes the input/ output pair h, z from the setup using a large random exponent α to compute $\hat{h} = h^\alpha \pmod N$ and $\hat{z} = z^\alpha \pmod N$. The committer computes a hash of the output (and public parameters pp) to generate a key $k = \mathsf{H}(\hat{z}, pp)$ which is used to encrypt the bid b with the opening proof $\pi_{\mathsf{Open},C}$ using a CCA-secure symmetric encryption scheme, so $ct \leftarrow \mathsf{CCA.Enc}(k, (b, \pi_{\mathsf{Open},\mathsf{C}}))$. The timed trapdoor consists of the randomized repeated squaring input $\hat{h}$ and the ciphertext ct. So, to compute the bid and opening, it suffices to compute $\hat{z} = \hat{h}^{2^t} \pmod N$, the corresponding key $k = \mathsf{H}(\hat{z}, pp)$, and then decrypt the ciphertext ct.

```
TTD.Setup($\lambda, t$)
$pp_{\mathsf{C}} \leftarrow\!\$\ \mathsf{C.Setup}(\lambda)$
$(q_1, q_2, N, \mathbb{G}, g, h) \leftarrow\!\$\ \mathsf{RSAGGen}(\lambda)$
$z \leftarrow h^{(2^t \bmod \varphi(N))} \pmod N$
Return $pp = (pp_{\mathsf{C}}, N, h, z, t)$

TTD.Comm$^{pp}(b)$
$(com_{\mathsf{C}}, \pi_{\mathsf{Open},\mathsf{C}}) \leftarrow\!\$\ \mathsf{C.Comm}(b)$
$\alpha \leftarrow\!\$\ [2^{2\lambda}]$
$\hat{h} \leftarrow h^\alpha \pmod N$ ; $\hat{z} \leftarrow z^\alpha \pmod N$
$k \leftarrow \mathsf{H}(\hat{z}, pp)$
$ct \leftarrow\!\$\ \mathsf{CCA.Enc}(k, (b, \pi_{\mathsf{Open},\mathsf{C}}))$
Return $(com = (com_{\mathsf{C}}, (\hat{h}, ct)), \pi_{\mathsf{Open}} = (\mathtt{committer}, \alpha))$

TTD.ForceOpen$^{pp}(com = (com_{\mathsf{C}}, (\hat{h}, ct)))$
$\hat{z} \leftarrow \hat{h}^{2^t} \pmod N$
$\pi_{\mathsf{PoE}} \leftarrow\!\$\ \mathsf{PoE.Prove}(N, \hat{h}, \hat{z}, t)$
$k \leftarrow \mathsf{H}(\hat{z}, pp)$
$(b, \pi_{\mathsf{Open},\mathsf{C}}) \leftarrow \mathsf{CCA.Dec}(k, ct)$ ; $\pi_{\mathsf{Open}} \leftarrow (\hat{z}, \pi_{\mathsf{PoE}})$
If not $\mathsf{C.VerOpen}(com_{\mathsf{C}}, b, \pi_{\mathsf{Open},\mathsf{C}})$ then return $(\perp, (\mathtt{force}, \pi_{\mathsf{Open}}))$
Else return $(b, (\mathtt{force}, \pi_{\mathsf{Open}}))$

TTD.VerOpen$^{pp}(com = (com_{\mathsf{C}}, (\hat{h}, ct)), b, \pi_{\mathsf{Open}} = (\mathtt{mode}, \pi))$
If $\mathtt{mode} = \mathtt{force}$
    Parse $\pi = (\hat{z}, \pi_{\mathsf{PoE}})$
    $k \leftarrow H(\hat{z}, pp)$ ; $(b', \pi_{\mathsf{Open},\mathsf{C}}) \leftarrow \mathsf{CCA.Dec}(k, ct)$
    Return 0 if $\mathsf{PoE.Ver}((N, \hat{h}, \hat{z}, t), \pi_{\mathsf{PoE}}) = 0$ or $b \neq b'$
Else if $\mathtt{mode} = \mathtt{committer}$
    Parse $\pi = \alpha$ ; $\hat{z} \leftarrow z^\alpha \pmod N$
    $k \leftarrow H(\hat{z}, pp)$ ; $(b', \pi_{\mathsf{Open},\mathsf{C}}) \leftarrow \mathsf{CCA.Dec}(k, ct)$
    Return 0 if $\hat{h} \neq h^\alpha \pmod N$ or $b \neq b'$
If $b = \perp$, return $(\mathsf{C.VerOpen}(com_{\mathsf{C}}, b, \pi_{\mathsf{Open},\mathsf{C}}) = 0)$
Else if $b \neq \perp$, return $(\mathsf{C.VerOpen}(com_{\mathsf{C}}, b, \pi_{\mathsf{Open},\mathsf{C}}) = 1)$
```

Figure 1: A non-malleable timed commitment TTD, parameterized by a non-malleable (non-timed) commitment scheme C, a proof of exponentiation PoE, and a CCA-secure symmetric encryption scheme CCA. The construction is in the random oracle model where all algorithms have access to the hash function H, modeled as a uniformly random function initialized during setup.

Additionally, we note that if a committer deviates from the protocol and produces an invalid commitment, i.e., a commitment in which either (1) the key derived from $k = \mathsf{H}(\hat{z}, pp)$ fails to decrypt ct, or (2) the decrypted opening $(b', \pi_{\mathsf{Open},\mathsf{C}}) \leftarrow \mathsf{CCA.Dec}(k, ct)$ fails to open com_{C}, force-opening will return $\perp$ along with a proof that the commitment is malformed. A proof of exponentiation (e.g. via the protocol of Wesolowski [Wes19] or Pietrzak [Pie19]) is computed to convince the verifier of the correct computation of output $\hat{z}$; given this element, the verifier can confirm for themselves that and the timed trapdoor is invalid.

The full details of the timed commitment protocol are provided in Figure 1, and we defer the proofs of security to Appendix A.2. Security relies on a trusted setup to compute an RSA group of hidden order. However, given such a setup, the RSA group can be reused across many delay parameter configurations. To use a new delay parameter t, a (h, z) pair where $z = h^t \pmod N$ must be computed and included as part of the public parameters. Without the RSA group trapdoor, computing such a pair will take time on the

Protocol: Sealed-Bid Auction with Timed Commitments

Initialization: The auction is initialized with public parameters for a timed commitment scheme C and delay parameter t.

Phase 1: Bid collection

(1) The auctioneer starts accepting bids at time t_0.
(2) To place a bid, a user must:
   (a) Commit to bid b, sending com to the auctioneer, $(com, \pi_{\mathsf{Open}}) \leftarrow\$ \mathsf{C.Comm}(b)$. User stores opening π_{Open}.
   (b) Lock up bid amount with the auctioneer (see Figure 3). Lock up opening rewards with the auctioneer in the amount of rwd_{Open} and rwd_{Force}.
(3) The auctioneer ends bid collection at time $t_0 + t$.

Phase 2: Bid self-opening

(1) Users provide openings π_{Open} computed in phase 1 to reveal bid b.
(2) Auctioneer verifies opening ($\mathsf{C.VerOpen}(com, b, \pi_{\mathsf{Open}})$). If opening is valid, rewards rwd_{Open} and rwd_{Force} are unlocked, and the bid entry is marked as opened.
(3) At time $t_0 + t + t_{\mathsf{Open}}$, any bid entries that were not self-opened are marked as abandoned. The locked up reward rwd_{Open} is forfeited.

Phase 3: Bid force-opening

(1) Each abandoned bid is force-opened by a third-party opener, $(b, \pi_{\mathsf{Open}}) \leftarrow \mathsf{C.ForceOpen}(com)$.
(2) Auctioneer verifies opening ($\mathsf{C.VerOpen}(com, b, \pi_{\mathsf{Open}})$). If opening is valid, the opener receives rwd_{Force}. If $b \neq \perp$, the bid entry is marked as opened, else it is marked as invalid.
(3) Auctioneer ends bid force-opening when there are no more abandoned entries.

Output: Auction results are determined from the bids marked as opened.

Figure 2: Sealed-bid auction protocol of Riggs-RP **and** Riggs-TC **proceeds in phases. The highlighted integration of timed commitments is included only in** Riggs-TC**.**

order of the delay parameter, but once computed, it can be reused to compute any number of commitments. The party computing pair (h, z) may also compute a proof of exponentiation to allow for others to verify the wellformedness of the public parameters.

5 DECENTRALIZED SEALED-BID AUCTIONS

Here we present our two auction protocols, Riggs-RP and Riggs-TC. Riggs-RP uses range proofs to verify validity of bids across asynchronous concurrent auctions. Riggs-TC extends Riggs-RP with timed commitments to protect against DoS by bidders who refuse to open their bids (§3).

5.1 Range Proofs for Concurrent Auctions

In Riggs-RP, each user is associated with a collateral that is used to back all bids to auctions a user has participated in. Auctions consist of two phases (detailed in Figure 2). First, in the bid collection phase, users choose bid b, generate a Pedersen commitment $com_{\mathsf{Ped}} = g^b h^{\alpha_b}$, and lock up a portion of their collateral (we describe the lock-up mechanism below). In the second phase, users open and reveal their committed bid, and the results of the auction are determined from the opened bids. If a user does not open their commitment, they forfeit the locked-up portion of their collateral.

As discussed in Section 3.2, when sharing a single collateral pool across many auctions, new bids and changes to the collateral pool must not invalidate outstanding bids. Riggs-RP enforces this by requiring a user to prove statements about the collateral and a user's active bids (using range proofs) whenever they bid or change their collateral pool. In particular, Riggs-RP stores a user's collateral bal, plus a commitment to the sum of the user's active bids com_{active}.

- When a user places a new bid, they prove that their bid is bounded ($0 \leq b < 2^{32}$) and that the sum of their current active bids B and new bid is at most their collateral ($B + b \leq bal$). If these range proofs verify, the bid commitment is accepted and com_{active} is updated by summing the bid commitment homomorphically.
- When a user wishes to withdraw part of their collateral, they prove that the amount being withdrawn amt does not cause the remaining collateral to not cover active bids, i.e., $B < bal - amt$.

The details of the auction house protocol are given in Figure 3.

In Figure 3, for simplicity of presentation, we overload use of the delay parameter t both as the cryptographic delay parameter and as a unit of time. In practice, determining the relationship between the cryptographic delay parameter and the wall-clock time delay is an intricate process; we discuss this further in Section 7.

5.2 Timed Commitments for DoS Protection

As discussed in Section 3.2, Riggs-RP relies on each bidder to come online and open their bid commitment. To ensure that all bids are opened, Riggs-TC uses our timed trapdoor commitment (§4) alongside the Pedersen commitment to the bid amount used in Riggs-RP. Riggs-TC also adds a force-opening phase after the self-opening phase concludes; any bid commitments that have not yet been opened are forced open using the timed trapdoor, and the auction results are not determined until all bids have been opened. This means that bidders cannot abandon their bids, nor are they incentivized to DoS other bidders in order to exclude bids from the final result. Figure 2 details the force-opening phase.

Incentives for force opening. Force-opening entails a significant amount of sequential work; this work (and thus the auction) will not be completed without an incentive. Both the seller and honest bidders have an incentive to force-open bids to complete the auction and unlock collateral, but relying on these parties to force-open may affect the efficiency of the auction: rational bidders will price in the expected force-opening cost when placing their bids. Worse, the total cost of force-opening can be manipulated, e.g., by malicious bidders who submit and abandon low-value bids.

To address this, Riggs-TC requires bidders to lock-up a reward for the first party who forces open their abandoned bid. This reward must be chosen to closely match the cost of opening an abandoned bid: too small and abandoned bids will never be opened, too large and adversaries are incentivized to DoS honest bidders and then force-open their bids. One potential solution is to choose the reward and delay parameters by consulting a marketplace like OpenSquare [TGB+21]. Even so, introducing a force-opening reward without care enables certain types of undesireable behavior such

Protocol: Asynchronous Auction House with Timed Commitments

Initialization: The auction house is initialized with public parameters for the Ped commitment scheme and for the TTD timed commitment scheme. The auction house stores a list of active auctions *active* in addition to the following for each associated user:

- *bal*: Account collateral balance of the user.
- com_{active}: Pedersen commitment to sum of active bids of the user.

Since the auction house is asynchronous, the protocol is not defined by phases, but rather by asynchronous actions.

Action 0: Auction registration: A seller may register an auction with the auction house to start any time t_0, specifying the length of the auction with a time parameter t. The seller will also provide the necessary additional public parameters for TTD for t. The auction house immediately begins an instance of the auction protocol from Figure 2 adding it to *active*.

Action 1: Balance updates

(1) A user may deposit funds into their account. The auction house updates the collateral *bal* accordingly, $bal \leftarrow bal + amt$.

(2) A user may withdraw funds from their account by providing a proof for the following relation where *amt* is the amount being withdrawn:

$$\mathcal{R}_{\mathsf{wdrw}} = \left\{ \begin{array}{c} \Big(\ (com_{active}, bal, amt), (B, \alpha) \ \Big) : \\ com_{active} = g^B h^\alpha \wedge 0 \le B \le bal - amt \end{array} \right\} .$$

If the proof verifies, the amount is withdrawn and the user's collateral is updated.

Action 2: Bidding
Users may place a bid on any auction that is in its bid collection phase (see Figure 2). To place a bid:

(1) A user locks up the bid amount and requisite opening rewards by producing a range proof. The user collateral is updated accordingly: $bal \leftarrow bal - rwd_{\mathsf{Open}} - rwd_{\mathsf{Force}}$. A user submits a commitment to their bid com_{Ped} (or timed commitment $com_{\mathsf{TTD}} = (com_{\mathsf{Ped}}, com_t)$) and proves:

$$\mathcal{R}_{\mathsf{bid}} = \left\{ \begin{array}{c} \Big(\ (com_{\mathsf{Ped}}, com_{active}, bal), (b, B, \alpha_b, \alpha_B) \ \Big) : \\ com_{\mathsf{Ped}} = g^b h^{\alpha_b} \wedge com_{active} = g^B h^{\alpha_B} \\ \wedge\ 0 \le b \le 2^{32} \ \wedge\ B + b \le bal \end{array} \right\} .$$

(2) If the proof verifies, the bid is accepted for the auction, and the user's active bids commitment is updated,
$com_{active} \leftarrow com_{active} \cdot com_{\mathsf{Ped}}$.

Action 3: Auction results
When an auction ends, the auction is removed from *active* and the results are incorporated into the auction house state:

(1) The value of each opened bid b is removed from the user's active bid commitment, $com_{active} \leftarrow com_{active}/g^b$.

(2) Any bid amount $amt \le b$ determined to be transacted as part of the auction results is subtracted from the user's collateral and deposited to the seller, $bal \leftarrow bal - amt$.

Figure 3: Asynchronous auction house protocol of Riggs-RP **and** Riggs-TC **to handle multiple concurrent auctions. The highlighted integration of timed commitments is included only in** Riggs-TC**.**

as griefing and front-running attacks. We address each of these attacks in Riggs-TC in turn below.

The bidder themself (who knows the opening to their own commitment) has a small advantage in computing a force-opening proof for their commitment, compared to a party with no such knowledge [Wes19]. They also have the opportunity to begin computing the force-opening proof early (before their bid is publicly available). This enables a griefing attack: a bidder pretends to abandon their bid, waits for others to do most of the force-opening work, then announces the force-opening proof, thus retrieving the reward and wasting others' time. To disincentivize this behavior, Riggs-TC also requires bidders to lock-up a separate self-opening reward that is forfeit if a bidder abandons their bid, ensuring that users lose money on net from griefing. Figures 2 and 3 give details.

Finally, we must consider front-running attacks. Here, a malicious party does no work but attempts to intercept and replay the force-opening proof of an honest party to claim the reward for themself. To solve this, we can use *watermarked proofs* [KOR04, Wes19, ABC22] which allow the prover to embed associated data that is authenticated during proof verification. In our timed commitment protocol, force-opening requires producing a proof of integer exponentiation [Wes19] (see Figure 1). An honest party embeds their identity as the watermark for the proof of integer exponentiation. The watermark is verified as part of the force-opening proof and the opening reward is given to the watermarked identity. Watermarking prevents front-running attacks: since malicious parties are not able to tamper with watermarked proofs to edit the watermark (e.g., to change the watermark to a different identity), only a party that performed the sequential work needed to produce a valid watermarked proof will be eligible for the reward. Supporting watermarks in existing proofs of integer exponentiation approaches incurs minimal overhead, simply requiring committing to the watermark in the challenge-generation step of the Fiat-Shamir proof [Wes19, ABC22].

5.3 Security

We now consider how the proposed construction, Riggs-TC meets the security goals of the auction setting. We consider Riggs-TC as it is a superset of the functionality provided by Riggs-RP. We discuss the goals of completeness and bid privacy.

Completeness. First consider the *completeness* of the construction. An auction that is complete should (1) result in a winner and price determined by the auction protocol (e.g., second-price) among all valid bids, and (2) result in the winning bidder paying the seller the appropriate winning price. Riggs-TC has two mechanisms for determining the correct winner and price of the auction. First, the binding property of the (timed) commitment scheme ensures that it is infeasible for bids to be opened to anything other than one value; this prevents malicious bidders from equivocating on their bid. Second, the completeness property of the timed commitment scheme means that commitments are guaranteed to be opened to their one value even without further interaction from the bidder; this prevents honest bidders from being excluded from the final auction results. Once a winner and price is determined, Riggs-TC relies on the soundness of the range proofs along with the additive homomorphic properties of the Pedersen commitment to ensure auction payment. By construction of the protocol, a Pedersen commitment to the sum of all active bids is maintained using the additive homomorphic property of Pedersen commitments. Finally, from the soundness of the range proof protocol for relations $\mathcal{R}_{\mathsf{wdrw}}$ and $\mathcal{R}_{\mathsf{bid}}$ (see Figure 3), the sum of all active bids for a bidder is always less than or equal to the bidder's collateral, thereby guaranteeing the ability to payout in the event of a win.

Bid privacy. In addition to completeness which reasons about the completion of an auction, we consider *bid privacy* in which we reason about the confidentiality of bid values during the bidding phase. Recall that the incentive compatibility of sealed-bid auctions relies on the inability of bidders to see the bids of others before placing their own bid. Consider again the strawman proposal in which a per-auction collateral is locked up (see Section 3.2). Here, the argument is simple: the bid commitment completely hides the bid value by the hiding property of the commitment, thus the only information leaked is from the public collateral in that a valid bid must fall below the collateral value. We depart from this strategy in favor of a shared collateral across all auctions that a bidder is participating in simultaneously. Our approach has usability benefits in that bidders may more easily participate in simultaneous auctions without locking up excessive amounts of funds but comes at the cost of a weaker bid privacy guarantee.

More precisely, Riggs-TC (and Riggs-RP) necessarily leak an upper bound of the sum of a bidder's bids across all active auctions as the bidder's public collateral. Further, observing this leakage over time as auctions begin and end (and over a possibly changing collateral) may allow for more fine-grained inference on a bidder's bids for long-term auctions. Note that this leakage is necessary to achieve the completeness property described above and ensure winner payout. Ultimately, the flexibility of a shared collateral allows a bidder some plausible deniability over how their bids are distributed across their active auctions, but it does not provide a shortcut to providing full bid privacy with respect to the maximum bid amount for each auction. For that, the bidder would need to incur the large collateral lock-up of the per-auction strawman. We leave to future work, an empirical analysis of the leakage and efficacy of inference attacks in realistic simultaneous auction scenarios. Lastly, an adversary does not learn anything more beyond the above described leakage. In Riggs-TC, we appeal to the functional non-malleability of the timed commitment (analog to hiding in the non-timed setting).

Formal analyses. As mentioned above, we provide formal security definitions for timed commitments and proofs of security for our proposed timed commitment primitive TTD, the core underlying component of Riggs-TC (deferred to Appendix A). This formal treatment does not cover the full auction protocol informally discussed above. Developing formal models suitable for analysis of these higher level primitives remains an open problem.

6 IMPLEMENTATION

We implement our constructions in Rust. Our auction house implementation consists of a number of modular libraries that may be of independent interest. We implement a Bulletproofs range proof library in the arkworks ecosystem (for pairing-based cryptography and zero-knowledge proofs) [BCG+20]. The implementation is agnostic to the choice of curve, however our evaluation is performed over the BN254 curve [BN05], which, looking forward, will be efficient within the Ethereum Virtual Machine (EVM). We also implement a proof of exponentiation for RSA hidden order groups [Wes19]; we discuss several implementation optimizations (for hash-to-prime) to reduce EVM costs. Lastly, we provide a timed commitments library of the constructions from Section 4 as well as a SNARK-based timed trapdoor (used as a baseline). In evaluation, we instantiate the timed commitments (and proof of exponentiation) with an RSA group of 2048 bits. Putting these libraries together, we implement the Riggs-RP and Riggs-TC auction house protocols. The protocols are described for any single-round sealed-bid auction; we implement a second-price auction. In total, our Rust implementation consists of $\approx$ 7000 lines of code and is available open source [2].

Ethereum smart contracts. In addition to evaluating our auction house through a Rust implementation, we implement an auction house smart contract in Solidity that can be deployed on the Ethereum blockchain. The smart contract is compatible with the Rust implementation, i.e., client operations are computed using the Rust library and the produced outputs are serialized into a contract call. Our Ethereum auction house smart contract integrates with existing Ethereum standards. An auction can be created for any non-fungible token (NFT) that follows the ERC-721 standard, and the auction house collateral is made up of an "auction house token" that abides by the ERC-20 standard for fungible tokens.

Our auction protocols additionally require an "auction completion" contract call to compute the auction winner, complete the payout between the winner(s) and seller, and refund the losers. Looking forward, we find that the cost of this contract call scales with the number of bidders in the auction, so it is not sufficient for it to be paid solely by the seller. Instead, we have bidders escrow some funds with the contract to reimburse the caller of the completion call; the cost of the contract call varies with the price of gas (Ethereum's computation unit), to address this variance, one may escrow gas tokens instead of ether [BDTJ18]. In total, our smart contracts implementation consists of $\approx$ 2000 lines of Solidity.

Hash-to-prime optimization. The dominating cost in verification of the proof of exponentiation (for force-opening) is validation of the hash-to-prime. The non-interactive proof of exponentiation requires a prime challenge drawn from a space of size twice the security parameter (i.e., 256 bits of entropy for 128 bits of security) [BBF18]. Typically, testing primality of a large prime using Miller-Rabin primality test is concretely efficient but is expensive in the EVM computation model. Instead, we use rejection sampling on carefully constructed integers to find a prime that admits generation of a short Pocklington primality certificate [BLS75]; the certificate can be efficiently verified with the EVM. Our techniques are related to those applied in other computation models such as embedded systems [CFTP12] and zero-knowledge constraint systems [OWWB20].

[2] https://github.com/nirvantyagi/riggs

Baselines. To evaluate our proposed constructions, we additionally implement two baselines. We implement the per-auction collateral solution described in Section 3.2 as a minimal comparison point with limited features. We also implement a SNARK-based solution with the same feature set as Riggs-TC. The SNARK construction is the same as Riggs-TC except we replace the timed trapdoor of Section 4 with a SNARK proof of equivalence between the Pedersen commitment and HTC timed commitment. We implement the SNARK timed commitment in Rust generic to the choice of SNARK proving system and pairing-friendly curve within arkworks, and evaluate using the Groth16 [Gro16] proof system. We also provide smart contract implementations in Solidity for the baselines, including SNARK verifier implementations for Groth16. We instantiate the Pedersen commitment over the Edwards curve on BN254 (i.e., Baby Jubjub) which admits efficient circuit encodings when proved with a SNARK over BN254. However, Baby Jubjub does not admit the same efficiency benefits of BN254 on EVM; it does not have precompiled opcodes for group operations.

7 EVALUATION

This section answers the following questions:

- What are the costs to host an auction?
- What are the costs for a bidder to participate?
- What are the costs of force-opening and what are the implications for setting time parameters?

To answer these questions, we must also evaluate the costs for the participants (verifiers or miners) in the decentralized consensus protocol. In our auction protocols, consensus participants are tasked with, for example, verifying commitment openings and maintaining collateral balances. The cost of performing these tasks is passed on to the bidders and seller in the form of transaction fees.

We evaluate blockchain costs in two ways (shown in Figure 4). First, we provide cost as the running time of our Rust implementation, which represents the case of running a special-purpose blockchain dedicated to hosting auctions. Second, we provide the costs of running our protocols as a smart contract on the Ethereum blockchain. These costs are measured in *gas*, a currency used in Ethereum to assign cost to operations in the EVM computational model; we report gas costs per the London hard fork of Ethereum.[3]

The blockchain costs can be divided into costs that are fixed per-auction and costs that scale linearly with the number of participating bidders. In our smart contract implementation, we require bidders to pay the per-bidder cost and the seller to pay the fixed costs. but other payment configurations are possible (e.g., winner pays fixed costs). A bidder also incurs costs to compute commitments and range proofs; we report the running times in Figure 5. All benchmarks for bidders and consensus participants were performed using an Intel Core i7-1165G7 processor with 4 cores.

Fixed costs to host an auction. To host an auction, two fixed costs that edit blockchain state and require blockchain consensus are incurred: auction creation and auction completion. In auction creation, the seller initializes new state on the blockchain for the new auction, incurring a cost that depends on the blockchain's storage model. For Ethereum, the per-auction collateral baseline and Riggs-RP have the smallest costs, followed by Riggs-TC and the SNARK baseline, which are ≈2.5× greater. This is because the latter protocols need extra storage for timed commitment parameters.

Completing an auction consists of (1) a fixed-size computation that pays from winner(s) to seller, and (2) a computation that scales with the number of bidders, namely, computing the winner and updating each bidder's state based on the result of the auction. The fixed-size computation is minimal for all of our auction protocols; we discuss per-bidder costs next.

Per-bidder costs to participate in an auction. A bidder incurs cost for (1) submitting a bid to the blockchain, (2) opening their bid, and (3) computing their portion of the completion procedure. (We discuss force-opening further below.)

To submit a bid, a bidder must first perform some local computation to prepare their bid and accompanying proof material, then pay for the blockchain computation that stores the bid and verifies the proofs. Figure 5 shows the running time for the local computation (no proof is required for the per-auction collateral baseline). Riggs-RP and Riggs-TC require two range proofs, which are the dominant cost; Riggs-TC is slightly more expensive since it also requires computing the timed commitment. The SNARK baseline requires computing a proof of equivalence between the Pedersen commitment and the timed commitment; this cost dominates, resulting in $\approx 300\times$ overhead. We discuss on-blockchain costs below.

Self-opening costs are as follows; note that the bidder has already computed the opening during bid submission, so no further local computation is required. The per-auction collateral baseline admits the cheapest verification: because it does not use any algebraic proofs, it uses a hash-based commitment that is cheap to verify on-chain. The other three protocols require opening a Pedersen commitment; while costlier than hashing, this is still relatively inexpensive.

Finally, for Riggs-RP, Riggs-TC, and the SNARK baseline (which maintain a unified per-user collateral instead of collateral per user per auction), a bidder may also deposit and withdraw funds from their collateral. Deposits are essentially free, while withdrawals require submitting a range proof showing that the withdrawal doesn't reduce the collateral below the sum of the user's active bids; the cost of computing the withdrawal proof is small (<50 ms).

Blockchain costs for submitting a bid. Figure 4 shows on-chain costs for submitting a bid. The per-auction collateral baseline simply stores the commitment and does not require proof verification. Riggs-RP and Riggs-TC entail range proof verification, and the SNARK baseline requires verification of the SNARK. We discuss these costs below.

Riggs-RP and Riggs-TC have rather high per-bid costs: at time of writing, 2.5 million gas costs ≈$134 on Ethereum. In practice, we believe the cost (in dollars) of running auctions would be much lower, for two reasons. First, EVM-compatible chains like EOS [eos] and Avalanche [ava] are growing rapidly in popularity, and have orders of magnitude lower transaction costs (concretely, the cost of a bid on Avalanche at time of writing would be ≈$1.50). Second, ≈90% of per-bid cost is verifying Bulletproofs; since this computation is useful elsewhere (e.g., for private payments [BAZB20, Dia21]), it

[3]Concretely, one unit of Ethereum gas costs $\approx\$10^{-4}$ at time of writing.

		Per-auction col.		Riggs-RP		Riggs-TC		SNARK	
Auction phase	cost type	native (µs)	EVM (gas)	native (µs)	EVM (gas)	native (µs)	EVM (gas)	native (µs)	EVM (gas)
Auction creation	fixed	-	23×10^4	-	25×10^4	-	71×10^4	-	74×10^4
Bid collection	per-bidder	0.7 (0.5)	16×10^4	2100 (200)	191×10^4	2100 (200)	221×10^4	8100 (400)	10243×10^4
Bid self-opening	per-bidder	1.6 (0.3)	10×10^4	210 (30)	10×10^4	220 (20)	10×10^4	300 (10)	148×10^4
Auction completion	fixed	3 (1)	12×10^4	2.9 (0.4)	12×10^4	3.1 (0.4)	12×10^4	2.5 (0.4)	12×10^4
	per-bidder	0.3 (0.2)	6×10^4	1.3 (0.2)	9×10^4	1.1 (0.3)	10×10^4	3.4 (0.3)	11×10^4

Figure 4: Costs for each phase of an auction, given both as a running time (µs) of our native Rust implementation and as a gas cost for the EVM compilation of the corresponding computation. The costs for each phase are specified as fixed per-auction or linearly scaling per-bidder. The running time is given with a standard deviation while the gas cost is deterministically computed.

Operation	Per-auction col.	Riggs-RP	Riggs-TC	SNARK
Submit bid	0.0006 (0.0001)	66 (1)	71 (3)	21700 (700)
Withdraw	-	34 (2)	34 (2)	34 (2)

Figure 5: Bidder running time (ms) for computing commitments and proofs for bid submission and collateral withdrawal.

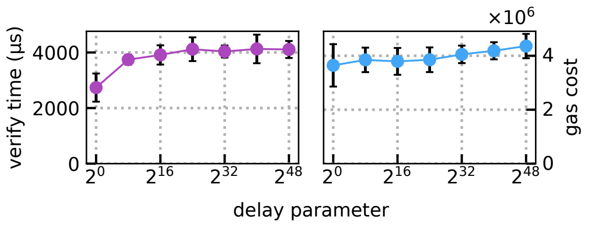

Figure 6: Verification costs of force-opening as a function of the delay parameter.

might eventually become a *precompiled contract* (i.e., a built-in primitive), slashing costs [4].

In addition to high monetary costs, throughput limitations of existing blockchain infrastructure are another barrier to the deployment of decentralized auctions. For example, Ethereum's current target block size is 15 million gas and block time is 12 seconds, meaning current throughput will only allow for < 50 bids to be collected per minute (in the optimistic case where full throughput is dedicated to auctions). Again, we expect scalability to improve as significant work is underway to improve scalability of layer 1 smart contract platforms as well as build layer 2 solutions (e.g., rollups) [But, WGH+, SSV21]. Our approach should extend to future systems as scalability improves. Future work may also consider special-purpose roll-up solutions tailored to the auction setting.

Estimating the delay parameter. To evaluate force-opening, we first need select the delay parameter t for each timed commitment, which is an intricate process when deploying time-based cryptographic tools. First, we must estimate the minimum desired wall-clock time below which no plausible adversary should be able to force-open a committed bid. In our auction setting, this time must be long enough to ensure no adversary can (privately) force-open a posted bid and still have time to post their own bid commitment, possibly influenced by learning one or more existing bids.

The bid collection phase can be relatively short: a sealed-bid auction does not require back-and-forth interaction as in an English auction; all bid commitments can be posted roughly simultaneously at auction start.[5] However, we must consider the maximum time between when a bidder broadcasts their bid to the network and when it is included in a block that is confirmed (with high probability) in the eventual longest chain. An attacker can begin force-opening as soon as a transaction is broadcast, and we can conservatively assume they can react to the forced-open bid and get their own bid included in block instantly. Transaction confirmation times vary based on network congestion and the gas price offered by the user. A common approximation is that Ethereum transactions are typically confirmed within 5 minutes.

Next, given a wall-clock time target such as 5 minutes, we must determine what delay parameter t in our timed commitment scheme achieves this minimum for plausible adversaries. This is also an imprecise process which requires assumptions about adversarial sequential computation speed. An attacker with an unlimited budget can likely eke out out marginally more computation speed. We can use existing best-in-class implementations to inform our assumptions: FPGAs have been shown to achieve 2^{24} modular squarings per second for a 2048 RSA modulus [Özt20], and speeds up to 2^{28} squarings per second are projected for ASICs with current technology [MÖS22]. Based on these results, we might take 2^{30} squarings per second as a conservative estimate of adversary capability. Combining this estimate with our 5 minute goal suggests a delay parameter $t = 2^{39}$.

We note that, compared to many other scenarios that use time-based cryptography, auctions are relatively robust to incorrect assumptions about attacker capabilities. First, delay parameters can be chosen close to the time of each auction, meaning there is no need to reason about future improvements in attacker speed. Second, the worst-case scenario of an attacker able to force-open bids in less than the desired time only reveals committed bids to the attacker. This does not help an attacker who is attempting to bid for and win the auction: they are still incentivized in the second-price format to bid their true valuation. The only attack this enables is for a seller observing committed bids to post a shill bid to raise the ultimate sale price, effectively reducing the auction to a first-price auction

[4]Our SNARK baseline takes advantage of Ethereum precompiled contracts purpose built for verifying SNARKs, which both skews the results and evidences an appetite for supporting such primitives.

[5]Since bidding in sealed-bid auctions requires no real-time human action, users who are not available at the designated bidding time, can set up a simple automated bidding agent to post their desired bid at the correct time.

or open-bid auction. The attacker cannot modify other users' bids or prevent the auction from completing.

Cost and time to force-open a bid. While the delay parameter is set based on conservative estimates of the best plausible adversary's capability, force-opening must be completed using whatever capability is available to honest parties.[6] Moreover, the cost of force-opening comprises both computing blinding factor of the timed commitment and proving the correct computation using a proof of exponentiation. Wesolowski [Wes19] provides an algorithm for computing the proof of exponentiation in $3 \cdot t/\log t$ group operations given storage of $\sqrt{t}$ group elements. With an ASIC performing the initial blinding factor computation and a CPU computing the proof of exponentiation, we would expect force-opening to take ≈ 8 hours.

In addition to locking up a reward to reimburse the force-opening party for the force-opening computation, the bidder also locks up a fee for blockchain costs associated with verifying force-opening (e.g., using gas tokens [BDTJ18] for Ethereum). Figure 6 shows how the verification cost of force-opening changes with the delay parameter. Asymptotically, the force-opening costs increase logarithmically with the delay parameter. However, in practice we find that the variance in hash-to-prime cost dwarfs the asymptotic growth. Verification costs remain low (< 5 ms) for large delay parameters, well beyond the conservative estimate for a 5 minute auction.

8 RELATED WORK

Auctions from timed commitments. Timed commitments have long been proposed (at least theoretically) for use with sealed-bid auctions [BN00, MT19, KLX20, FKPS21]. This work addresses the practical details of the deployment setting including handling concurrent auctions (with range proofs) and abandoned bids (with reward incentives), as well as optimizing and evaluating an implementation. Perhaps the most closely related work is that of Deuber et al. [DDM+20] that employs a similar "timed commitment to a commitment opening" approach for practical efficiency. In their setting, a first-price sealed-bid auction is proposed for minting new tokens where bids correspond to "waiting time", and the bidder willing to wait the longest wins the newly minted token. Here, bid currency corresponds to time, in contrast to tokens, and enforcing valid auction results using range proofs on token balances is not a concern; [DDM+20] also does not consider incentives for opening abandoned bids. Further, we formalize and prove that the strategy of generically combining a timed commitment with another commitment scheme results in a secure timed commitment with the desired properties (see Section 4 and Appendix A) – this modular result was not previously provided [DDM+20].

Sealed-bid auctions on the blockchain. A number of papers have proposed blockchain-based sealed-bid auction platforms, mostly building on top of Ethereum. These proposals can be divided into two broad approaches. The first category are protocols which use a variation of commit-reveal. These protocols generally guarantee bid privacy but cannot guarantee the auction will complete, relying on deposits and penalties to discourage participants from aborting [BCDB18, KST+20, LZL+21, CLXW22].

The second category of proposals utilize some form of off-chain auctioneer (or set of auctioneers) to collect and reveal bids. These protocols guarantee auction completion but rely on trust assumptions to ensure bid privacy. Many protocols assume a semi-trusted auctioneer, using a variety of cryptographic approaches to prove that the winner and winning price were computed correctly [GY18, DKK19, SVS+21, CC21, AMG21]. However, the auctioneer is trusted not to collude with bidders by revealing private bid values early. Other implementations use trusted enclaves [GY19, DK21] to fulfill the role of the auctioneer. Blass and Kerschbaum proposed STRAIN [BK18], in which all parties jointly implement the auction as a multi-party computation protocol, requiring an honest-majority assumption to prevent a malicious coalition from learning bid values early.

In contrast to all of these proposals, our protocols guarantee auction completion (as long as timed commitments can be force-opened) as well as bid privacy (as long as an adversary cannot exceed maximum estimated computation speeds).

The only concrete protocol we know of using timing assumptions is due to Xiong and Wang [XW19], using time-release encryption. However, their time-release encryption primitive relies on a trusted third party (the time server) rather than computational assumptions. Specifically, the time server releases information allowing the auctioneer to decrypt bids at the appropriate time. Thus the security model for bid privacy is similar to that in protocols with a semi-trusted auctioneer.

Finally, we note that we are unaware of any widely-deployed blockchain auction system in practice which uses sealed bids. Auctionity [LNPR18], for example, which bills itself as "the world's largest blockchain auction house for cryptocollectibles" uses an open-bid English auction format, as do major NFT platforms which implement auctions such as OpenSeas or Rarible. The traditional (offline) auction houses Sotheby's [sot21] and Christie's [chr21] both began auctioning NFTs with bids denominated in ETH in 2021, but maintained an English auction format with a fully trusted (and in fact, human) auctioneer.

Publicly verifiable secret sharing (PVSS). PVSS [Sta96] has been an alternative proposal to timed commitments for settings like sealed auctions that require unbiased outputs. Bids are verifiably secret-shared to a set of parties. Parties keep the secret share hidden during the duration of the auction so as to hide bids, and then communicate with other parties after the close of the auction to reconstruct bids. Auction integrity is maintained as long as some threshold of parties adhere to the protocol. We opt for timed commitments as they generally incur less communication costs and have a weaker trust model than PVSS-based approaches. Furthermore, timed commitments offer a route to a simpler blockchain-agnostic protocol, as opposed to PVSS which require determining a set of parties to secret share (which ideally would bootstrap on top of specific committee-based consensus protocols [GHM+17, DGKR18], e.g., [GKM+22, BGG+20, GHK+21, GHM+21]).

Distributed randomness beacons. Distributed randomness generation or beacons are protocols in which untrusted parties collaborate to produce an unbiased random output. Systems for distributed

[6] In the literature on VDFs the ratio of these speeds is often denoted $A_{\max}$.

randomness have thus been proposed based on PVSS [SJK$^+$17, CD17, SJSW20, BSL$^+$21, DKIR21] and timed commitments [LW15, BGB17, Dra18, SJH$^+$21, TCLM21]. Parties contribute randomness in the form of secret shares or timed commitment which are combined to form the final output. While the sealed-bid auction setting shares many similarities with distributed randomness generation, it introduces more complexities including verifying bid validity.

Front-running countermeasure for decentralized exchanges. Front-running, in the context of decentralized exchanges, is when privileged parties with low network latency (e.g., miners) observe incoming transactions, create new transactions of their own, and order transactions beneficially to claim the price difference value [DGK$^+$20] (see [BCD$^+$21] for detailed survey of front-running strategies). There is evidence that the existence of such privileged parties is inherent in the peer-to-peer network setting [TKFJ22], so instead commit-and-reveal protocols (that can be instantiated via PVSS or timed commitments) have been proposed to mitigate front-running by hiding the proposed transactions during a commit phase [ZMEF22, MGZ22]. Instantiating our sealed bid auction protocols with a double auction mechanism for matching buyers and sellers may result in a front-running resistant exchange protocol. We leave further evaluation of such a proposal to future work.

Delay encryption. Burdges and De Feo present a new primitive termed *delay encryption* related to timed commitments [BF21]. Given a fresh random identifier (e.g., from a randomness beacon), messages (bids) can be encrypted to the identifier such that the resulting ciphertexts can all be decrypted (opened) by computing a decryption key from the identifier with a sequential amount of work. Thus, only one sequential problem needs to be solved to open all bids, whereas with timed commitments, every abandoned bid requires a different sequential problem. Unfortunately, the only existing construction for delay encryption is based on isogenies and is not practical for deployment.

Related to delay encryption, there exists a line of work that aims to “encrypt to the future” bootstrapping on public values that are created during progression of the blockchain (e.g., signatures signed by trusted committee or sequence of solutions to computational puzzles). Using a primitive known as *witness encryption* [GGSW13], bids can be encrypted to a statement about these to-be-public values that can be decrypted when the witness (public value) is revealed in the future. Existing practical approaches [CDK$^+$21, DHMW22] have the limitation that they can only encrypt to a committee that is known ahead of time, which for relevant consensus protocols [GHM$^+$17, DGKR18] allows encrypting a few blocks ahead but not more than that, or using not yet practical witness encryption for NP [LJKW18].

Sealed payments on the blockchain. Zether [BAZB20, Dia21] and ZCash [zca] (among others) employ zero-knowledge proofs for hiding the amount transferred in a token transaction on the blockchain; Zether [BAZB20] employs range proofs in a manner similar to Riggs-RP to verify account balances, while ZCash proves the existence of an unspent transaction. Both works along with Zerocoin [MGGR13] also propose approaches for anonymizing the participants of a transaction. These techniques can be adapted to our setting to, if used with care [KYMM18], help hide collateral amounts (see Section 9). While these protocols, namely Zether, offer approaches for proving validity of hidden bids and collateral, they do not address denial-of-service attacks in the auction setting. In Riggs-TC, we show how to compose range proofs with timed commitments to mitigate such attacks.

Timed payments on the blockchain. A challenge that arises in contingent payments on the blockchain (e.g., multi-hop or cross-chain) is atomicity, enforcing that all dependent transactions either succeed or fail together. A hash time-lock contract (HTLC) [PD16] enables atomic transactions by requiring a recipient to acknowledge a payment by a certain time (typically determined by block numbers) or forfeit the ability to claim the transaction, returning the funds to the payer. HTLCs have been extended to hide the identity of participating parties [MMK$^+$17, MMS$^+$19, AEE$^+$21, TBM$^+$20] and for hiding other contingencies for transaction execution [Max15, CGGN17, BK19]. As in our setting, the time component requires thinking carefully about incentives for participating parties [TYME21]. However, HTLCs do not attempt to hide the payment amount which make them unsuitable for the sealed-bid setting without significant new machinery.

9 CONCLUDING DISCUSSION

We conclude by discussing several important directions for future work and extensions to the auction setting beyond the core protocols presented.

Multi-round auctions. It is straightforward to extend our protocols to multi-round auction formats such as the *simultaneous ascending auctions* employed by Federal Communications Commission for wireless spectrum auctions [GV99]. In such an auction, the public parameters (e.g., the reserve price or participant set) of the next auction is determined by the results of previous round auctions. Since the parameters are public, the blockchain can enforce the next round’s auction is parameterized appropriately.

Hidden collateral via private payments. The public collateral of a user serves as an upper bound for the sum of a user’s active bids. Thus, observing the collateral of a user leaks some information on the possible values of their sealed bids. This leakage can be reduced using techniques for private payments on the blockchain [BAZB20, zca]. Instead of storing public collateral, the auction house could store a commitment to collateral and support private payments between users’ collateral pools to further obfuscate collateral amounts. Private blockchain payment protocols are not perfect [KYMM18, GKRN18], but may be a worthwhile improvement over the current leakage in some settings.

Minimizing locked-up opening rewards. We can consider a hybrid protocol between Riggs-RP and Riggs-TC in which the timed commitment is optional. The auction protocol can allow users that are not worried about DoS attacks (e.g., large custodial “users” with significant infrastructure) to submit bids without timed commitments, omitting opening rewards. On a similar note, we can allow for bids to be submitted using varied delay parameters dependent on how close the auction is to the end of the bid collection phase. Bids submitted close to the end of the auction would allow for a smaller delay parameter and a correspondingly smaller opening reward lock-up.

Mitigating opportunity cost for participants. If a bid is abandoned, the duration of the auction (and time for which collateral is locked up) is extended while the force-opening phase completes. Participating bidders pay an opportunity cost in their locked up collateral, and the seller pays an opportunity cost in delayed payment for their locked up good. Some examples of possible mitigations might be releasing collateral lock-up early if a self-opened bid is not in contention to win (e.g., in a second price auction, if a bid is not the current highest bid), or allowing an auction to end without completing the force-opening phase if both seller and current winning bidder(s) agree on doing so. However, sellers opting for early completion may again incentivize bidders to mount denial-of-service attacks against each other. Another approach could be to compensate the seller and users with bids in contention with the forfeited self-opening rewards of abandoned bids. Of course, the opportunity cost is dependent on the amount of the bid, and since the bid amount is sealed, it seems difficult to set self-opening rewards in a way to guarantee to proper compensation. We leave further investigation to future work.

Credible auctions. Akbarpour and Li [AL20] introduce the notion of *credible* auctions, a game-theoretic property roughly stating that the auctioneer (or seller, in the decentralized setting) is disincentivized from deviating from the protocol even in undetectable ways. This includes, for example, submitting false bids under pseudonyms. Ferreira and Weinberg [FW20] show that, when assuming hiding and binding cryptographic commitments, it is possible to design credible sealed-bid second-price auctions by setting a high enough collateral to disincentivize abandoning bids while assuming certain properties about the distribution of bid values.

While these prior works analyze auctions in a setting with private communication, Chitra et al. [CFK23] show that the public broadcast channel of blockchains allow for credible auctions using much weaker assumptions about the distribution of bid values. A key requirement still is the sufficiently large upfront collateral bidders must pay, which is again calculated using assumptions on the distribution of bid values. However, as our auction protocols provide "efficient collateralization" by leveraging the linearity of the sealed-bid commitments to penalize abandoned bids for exactly the bid amount, they may provide credibility under weaker assumptions about the distribution of user bids. We leave a formal game-theoretic analysis of this as future work.

ACKNOWLEDGMENTS

Nirvan Tyagi was supported by NSF grant CNS-2120651 and by the Stanford Future of Digital Currency Initiative (FDCI); part of this work was completed while he was a visiting student at Stanford University. Cody Freitag's work was partially done while at NTT Research, and he is supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE–2139899, DARPA Award HR00110C0086, AFOSR Award FA9550-18-1-0267, and NSF CNS-2128519. Arasu Arun and Joseph Bonneau were supported by DARPA under Agreement No. HR00112020022 and by a16z crypto research. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Government, DARPA, a16z, or any other supporting organization.

REFERENCES

[ABC22] Arasu Arun, Joseph Bonneau, and Jeremy Clark. Short-lived zero-knowledge proofs and signatures. In *ASIACRYPT (3)*, volume 13793 of *Lecture Notes in Computer Science*, pages 487–516. Springer, 2022.

[ADMM14] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. Secure multiparty computations on bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 443–458. IEEE, 2014.

[AEE+21] Lukas Aumayr, Oguzhan Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostáková, Matteo Maffei, Pedro Moreno-Sanchez, and Siavash Riahi. Generalized channels from limited blockchain scripts and adaptor signatures. In *ASIACRYPT (2)*, volume 13091 of *Lecture Notes in Computer Science*, pages 635–664. Springer, 2021.

[AL20] Mohammad Akbarpour and Shengwu Li. Credible Auctions: A Trilemma. *Econometrica*, 88(2):425–467, March 2020.

[AMG21] Hussein Abulkasim, Atefeh Mashatan, and Shohini Ghose. Quantum-based privacy-preserving sealed-bid auction on the blockchain. *Optik*, 242, 2021.

[AN20] Ramiro Alvarez and Mehrdad Nojoumian. Comprehensive survey on privacy-preserving protocols for sealed-bid auctions. *Computers & Security*, 88:101502, 2020.

[ava] Avalanche. https://www.avalabs.org/.

[BAZB20] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. In *Financial Cryptography*, volume 12059 of *Lecture Notes in Computer Science*, pages 423–443. Springer, 2020.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society, 2018.

[BBF18] Dan Boneh, Benedikt Bünz, and Ben Fisch. A survey of two verifiable delay functions. *IACR Cryptol. ePrint Arch.*, page 712, 2018.

[BBF19] Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching techniques for accumulators with applications to iops and stateless blockchains. In *CRYPTO (1)*, volume 11692 of *Lecture Notes in Computer Science*, pages 561–586. Springer, 2019.

[BBHR19] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In *CRYPTO (3)*, volume 11694 of *Lecture Notes in Computer Science*, pages 701–732. Springer, 2019.

[BCD+09] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, et al. Secure multiparty computation goes live. In *International Conference on Financial Cryptography and Data Security*, pages 325–343. Springer, 2009.

[BCD+21] Carsten Baum, James Hsin-yu Chiang, Bernardo David, Tore Kasper Frederiksen, and Lorenzo Gentile. Sok: Mitigation of front-running in decentralized finance. *IACR Cryptol. ePrint Arch.*, page 1628, 2021.

[BCDB18] Chiara Braghin, Stelvio Cimato, Ernesto Damiani, and Michael Baronchelli. Designing smart-contract based auctions. In *International Conference on Security with Intelligent Computing and Big-data Services*, 2018.

[BCG+20] Sean Bowe, Alessandro Chiesa, Matthew Green, Ian Miers, Pratyush Mishra, and Howard Wu. ZEXE: enabling decentralized private computation. In *IEEE Symposium on Security and Privacy*, pages 947–964. IEEE, 2020.

[BCM05] Endre Bangerter, Jan Camenisch, and Ueli M. Maurer. Efficient proofs of knowledge of discrete logarithms and representations in groups with hidden order. In *Public Key Cryptography*, volume 3386 of *Lecture Notes in Computer Science*, pages 154–171. Springer, 2005.

[BDTJ18] Lorenz Breidenbach, Philip Daian, Florian Tramer, and Ari Juels. Gastoken: A journey through blockchain resource arbitrage. In *CESC*, 2018.

[BF21] Jeffrey Burdges and Luca De Feo. Delay encryption. In *EUROCRYPT (1)*, volume 12696 of *Lecture Notes in Computer Science*, pages 302–326. Springer, 2021.

[BGB17] Benedikt Bünz, Steven Goldfeder, and Joseph Bonneau. Proofs-of-delay and randomness beacons in ethereum. In *IEEE Workshop on Security and Privacy on the Blockchain*. IEEE Computer Society, 2017.

[BGG+20] Fabrice Benhamouda, Craig Gentry, Sergey Gorbunov, Shai Halevi, Hugo Krawczyk, Chengyu Lin, Tal Rabin, and Leonid Reyzin. Can a public blockchain keep a secret? In *TCC (1)*, volume 12550 of *Lecture Notes in Computer Science*, pages 260–290. Springer, 2020.

[BK18] Erik-Oliver Blass and Florian Kerschbaum. Strain: A secure auction for blockchains. In *European Symposium on Research in Computer Security*, 2018.

[BK19] Sergiu Bursuc and Steve Kremer. Contingent payments on a public ledger: Models and reductions for automated verification. In *ESORICS (1)*, volume 11735 of *Lecture Notes in Computer Science*, pages 361–382. Springer, 2019.

[BLS75] John Brillhart, Derrick H Lehmer, and John L Selfridge. New primality criteria and factorizations of $2^m \pm 1$. *Mathematics of computation*, 29(130):620–647, 1975.

[BN00] Dan Boneh and Moni Naor. Timed commitments. In *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 236–254. Springer, 2000.

[BN05] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer, 2005.

[Bou00] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444. Springer, 2000.

[BSL$^+$21] Adithya Bhat, Nibesh Shrestha, Zhongtang Luo, Aniket Kate, and Kartik Nayak. Randpiper - reconfiguration-friendly random beacons with quadratic communication. In *CCS*, pages 3502–3524. ACM, 2021.

[But] Vitalik Buterin. The dawn of hybrid layer 2 protocols. https://vitalik.ca/general/2019/08/28/hybrid_layer_2.html.

[CC21] Theodoros Constantinides and John Cartlidge. Block Auction: A general blockchain protocol for privacy-preserving and verifiable periodic double auctions. In *IEEE International Conference on Blockchain*, 2021.

[CCL$^+$20] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Bandwidth-efficient threshold EC-DSA. In *Public Key Cryptography (2)*, volume 12111 of *Lecture Notes in Computer Science*, pages 266–296. Springer, 2020.

[CCS08] Jan Camenisch, Rafik Chaabouni, and Abhi Shelat. Efficient protocols for set membership and range proofs. In *ASIACRYPT*, volume 5350 of *Lecture Notes in Computer Science*, pages 234–252. Springer, 2008.

[CD17] Ignacio Cascudo and Bernardo David. SCRAPE: scalable randomness attested by public entities. In *ACNS*, volume 10355 of *Lecture Notes in Computer Science*, pages 537–556. Springer, 2017.

[CDK$^+$21] Matteo Campanelli, Bernardo David, Hamidreza Khoshakhlagh, Anders K. Kristensen, and Jesper Buus Nielsen. Encryption to the future: A paradigm for sending secret messages to future (anonymous) committees. *IACR Cryptol. ePrint Arch.*, page 1423, 2021.

[CFK23] Tarun Chitra, Matheus V. X. Ferreira, and Kshitij Kulkarni. Credible, optimal auctions via blockchains, 2023.

[CFTP12] Christophe Clavier, Benoit Feix, Loïc Thierry, and Pascal Paillier. Generating provable primes efficiently on embedded devices. In *Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 372–389. Springer, 2012.

[CGGN17] Matteo Campanelli, Rosario Gennaro, Steven Goldfeder, and Luca Nizzardo. Zero-knowledge contingent payments revisited: Attacks and payments for services. In *CCS*, pages 229–243. ACM, 2017.

[CHJ$^+$20] HeeWon Chung, Kyoohyung Han, Chanyang Ju, Myungsun Kim, and Jae Hong Seo. Bulletproofs+: Shorter proofs for privacy-enhanced distributed ledger. *IACR Cryptol. ePrint Arch.*, page 735, 2020.

[CHM$^+$20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas P. Ward. Marlin: Preprocessing zksnarks with universal and updatable SRS. In *EUROCRYPT (1)*, volume 12105 of *Lecture Notes in Computer Science*, pages 738–768. Springer, 2020.

[chr21] Welcome to the future. Digital Art: NFTs. https://www.christies.com/auctions/christies-encrypted, Mar 2021.

[CKLR21] Geoffroy Couteau, Michael Klooß, Huang Lin, and Michael Reichle. Efficient range proofs with transparent setup from bounded integer commitments. In *EUROCRYPT (3)*, volume 12698 of *Lecture Notes in Computer Science*, pages 247–277. Springer, 2021.

[CLXW22] Biwen Chen, Xue Li, Tao Xiang, and Peng Wang. SBRAC: Blockchain-based sealed-bid auction with bidding price privacy and public verifiability. *Journal of Information Security and Applications*, 2022.

[CPP17] Geoffroy Couteau, Thomas Peters, and David Pointcheval. Removing the strong RSA assumption from arguments over the integers. In *EUROCRYPT (2)*, volume 10211 of *Lecture Notes in Computer Science*, pages 321–350, 2017.

[DDM$^+$20] Dominic Deuber, Nico Döttling, Bernardo Magri, Giulio Malavolta, and Sri Aravinda Krishnan Thyagarajan. Minting mechanism for proof of stake blockchains. In *ACNS (1)*, volume 12146 of *Lecture Notes in Computer Science*, pages 315–334. Springer, 2020.

[DF02] Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 125–142. Springer, 2002.

[DGK$^+$20] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *IEEE Symposium on Security and Privacy*, pages 910–927. IEEE, 2020.

[DGKR18] Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *EUROCRYPT (2)*, volume 10821 of *Lecture Notes in Computer Science*, pages 66–98. Springer, 2018.

[DHMW22] Nico Döttling, Lucjan Hanzlik, Bernardo Magri, and Stella Wohnig. Mcfly: Verifiable encryption to the future made practical. *IACR Cryptol. ePrint Arch.*, page 433, 2022.

[Dia21] Benjamin E. Diamond. Many-out-of-many proofs and applications to anonymous zether. In *IEEE Symposium on Security and Privacy*, pages 1800–1817. IEEE, 2021.

[DK21] Harsh Desai and Murat Kantarcioglu. SECAUCTEE: Securing Auction Smart Contracts using Trusted Execution Environments. In *2021 IEEE International Conference on Blockchain (Blockchain)*, 2021.

[DKIR21] Sourav Das, Vinith Krishnan, Irene Miriam Isaac, and Ling Ren. SPURT: scalable distributed randomness beacon with transparent setup. *IACR Cryptol. ePrint Arch.*, page 100, 2021.

[DKK19] Harsh Desai, Murat Kantarcioglu, and Lalana Kagal. A Hybrid Blockchain Architecture for Privacy-Enabled and Accountable Auctions. In *International Conference on Blockchain*, 2019.

[Dra18] Justin Drake. Minimal VDF randomness beacon. Technical report, Ethereum Research, 2018.

[eos] EOS. https://eos.io/.

[FKPS21] Cody Freitag, Ilan Komargodski, Rafael Pass, and Naomi Sirkin. Non-malleable time-lock puzzles and applications. In *TCC (3)*, volume 13044 of *Lecture Notes in Computer Science*, pages 447–479. Springer, 2021.

[FO97] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. Springer, 1997.

[FR96] Matthew K Franklin and Michael K Reiter. The design and implementation of a secure auction service. *IEEE Transactions on Software Engineering*, 22(5), 1996.

[FW20] Matheus V. X. Ferreira and S. Matthew Weinberg. Credible, truthful, and two-round (optimal) auctions via cryptographic commitments. In *EC*, pages 683–712. ACM, 2020.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *STOC*, pages 467–476. ACM, 2013.

[GHK$^+$21] Craig Gentry, Shai Halevi, Hugo Krawczyk, Bernardo Magri, Jesper Buus Nielsen, Tal Rabin, and Sophia Yakoubov. YOSO: you only speak once - secure MPC with stateless ephemeral roles. In *CRYPTO (2)*, volume 12826 of *Lecture Notes in Computer Science*, pages 64–93. Springer, 2021.

[GHM$^+$17] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *SOSP*, pages 51–68. ACM, 2017.

[GHM$^+$21] Craig Gentry, Shai Halevi, Bernardo Magri, Jesper Buus Nielsen, and Sophia Yakoubov. Random-index PIR and applications. In *TCC (3)*, volume 13044 of *Lecture Notes in Computer Science*, pages 32–61. Springer, 2021.

[GKM$^+$22] Vipul Goyal, Abhiram Kothapalli, Elisaweta Masserova, Bryan Parno, and Yifan Song. Storing and retrieving secrets on a blockchain. In *Public Key Cryptography (1)*, volume 13177 of *Lecture Notes in Computer Science*, pages 252–282. Springer, 2022.

[GKRN18] Steven Goldfeder, Harry Kalodner, Dillon Reisman, and Arvind Narayanan. When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencies. *PETS*, 2018.

[Goo23] Justice Department Sues Google for Monopolizing Digital Advertising Technologies. The United States Department of Justice, Jan 2023.

[GOP$^+$22] Chaya Ganesh, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. Fiat–shamir bulletproofs are non-malleable (in the algebraic group model). In *EUROCRYPT*. Springer, 2022.

[Gro05] Jens Groth. Non-interactive zero-knowledge arguments for voting. In *ACNS*, volume 3531 of *Lecture Notes in Computer Science*, pages 467–482, 2005.

[Gro11] Jens Groth. Efficient zero-knowledge arguments from two-tiered homomorphic commitments. In *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 431–448. Springer, 2011.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT (2)*, volume 9666 of *Lecture Notes in Computer Science*, pages 305–326. Springer, 2016.

[GT21] Ashrujit Ghoshal and Stefano Tessaro. Tight state-restoration soundness in the algebraic group model. In *CRYPTO (3)*, volume 12827 of *Lecture Notes in Computer Science*, pages 64–93. Springer, 2021.

[GV99] Sharon E Gillett and Ingo Vogelsang. *Competition, regulation, and convergence: current trends in telecommunications policy research*. Routledge, 1999.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *IACR Cryptol. ePrint Arch.*, 2019:953, 2019.

[GY18] Hisham S. Galal and Amr M. Youssef. Verifiable Sealed-Bid Auction on the Ethereum Blockchain. Cryptology ePrint Archive, Paper 2018/704, 2018.

[GY19] Hisham S Galal and Amr M Youssef. Trustee: Full Privacy Preserving Vickrey Auction on top of Ethereum. In *Financial Crypto*, 2019.

[KLX20] Jonathan Katz, Julian Loss, and Jiayu Xu. On the security of time-lock puzzles and timed commitments. In *TCC (3)*, volume 12552 of *Lecture Notes in Computer Science*, pages 390–413. Springer, 2020.

[KMS$^+$16] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE symposium on security and privacy (SP)*, pages 839–858. IEEE, 2016.

[KOR04] Jonathan Katz, Rafail Ostrovsky, and Michael O. Rabin. Identity-based zero knowledge. In *SCN*, volume 3352 of *Lecture Notes in Computer Science*, pages 180–192. Springer, 2004.

[KST$^+$20] Michal Król, Alberto Sonnino, Argyrios Tasiopoulos, Ioannis Psaras, and Etienne Rivière. PASTRAMI: privacy-preserving, auditable, Scalable & Trustworthy Auctions for multiple items. In *Proceedings of the 21st International Middleware Conference*, 2020.

[KYMM18] George Kappos, Haaroon Yousaf, Mary Maller, and Sarah Meiklejohn. An empirical analysis of anonymity in zcash. In *USENIX Security Symposium*, pages 463–477. USENIX Association, 2018.

[KZZ16] Aggelos Kiayias, Hong-Sheng Zhou, and Vassilis Zikas. Fair and robust multi-party computation using a global transaction ledger. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 705–734. Springer, 2016.

[Lip03] Helger Lipmaa. On diophantine complexity and statistical zero-knowledge arguments. In *ASIACRYPT*, volume 2894 of *Lecture Notes in Computer Science*, pages 398–415. Springer, 2003.

[LJKW18] Jia Liu, Tibor Jager, Saqib A. Kakvi, and Bogdan Warinschi. How to build time-lock encryption. *Des. Codes Cryptogr.*, 86(11):2549–2586, 2018.

[LNPR18] Pascal Lafourcade, Mike Nopère, Daniela Pizzuti, and Étienne Roudeix. Auctionity yellow paper. https://www.auctionity.com/wp-content/uploads/2018/09/Auctionity-Yellow-Paper.pdf, 2018.

[LR00] David Lucking-Reiley. Vickrey auctions in practice: From nineteenth-century philately to twenty-first-century e-commerce. *Journal of economic perspectives*, 14(3):183–192, 2000.

[LW15] Arjen K. Lenstra and Benjamin Wesolowski. A random zoo: sloth, unicorn, and trx. *IACR Cryptol. ePrint Arch.*, page 366, 2015.

[LZL$^+$21] Genhua Lu, Yi Zhang, Zhongxiang Lu, Jun Shao, and Guiyi Wei. Blockchain-based sealed-bid domain name auction protocol. In *EAI International Conference on Applied Cryptography in Computer and Communications*, 2021.

[Max15] Gregory Maxwell. Zero knowledge contingent payment. Technical report, Bitcoin Wiki, 2015.

[MGGR13] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *IEEE Symposium on Security and Privacy*, pages 397–411. IEEE Computer Society, 2013.

[MGZ22] Peyman Momeni, Sergey Gorbunov, and Bohan Zhang. Fairblock: Preventing blockchain front-running with minimal overheads. *IACR Cryptol. ePrint Arch.*, page 1066, 2022.

[MMK$^+$17] Giulio Malavolta, Pedro Moreno-Sanchez, Aniket Kate, Matteo Maffei, and Srivatsan Ravi. Concurrency and privacy with payment-channel networks. In *CCS*, pages 455–471. ACM, 2017.

[MMS$^+$19] Giulio Malavolta, Pedro Moreno-Sanchez, Clara Schneidewind, Aniket Kate, and Matteo Maffei. Anonymous multi-hop locks for blockchain scalability and interoperability. In *NDSS*. The Internet Society, 2019.

[MÖS22] Ahmet Can Mert, Erdinç Öztürk, and Erkay Savas. Low-latency ASIC algorithms of modular squaring of large integers for VDF evaluation. *IEEE Trans. Computers*, 71(1):107–120, 2022.

[MT19] Giulio Malavolta and Sri Aravinda Krishnan Thyagarajan. Homomorphic time-lock puzzles and applications. In *CRYPTO (1)*, volume 11692 of *Lecture Notes in Computer Science*, pages 620–649. Springer, 2019.

[NPS99] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 129–139, 1999.

[NS93] Hannu Nurmi and Arto Salomaa. Cryptographic protocols for vickrey auctions. *Group Decision and Negotiation*, 2(4), 1993.

[OWWB20] Alex Ozdemir, Riad S. Wahby, Barry Whitehat, and Dan Boneh. Scaling verifiable computation using efficient set accumulators. In *USENIX Security Symposium*, pages 2075–2092. USENIX Association, 2020.

[Özt20] Erdinç Öztürk. Design and implementation of a low-latency modular multiplication algorithm. *IEEE Trans. Circuits Syst. I Regul. Pap.*, 67-I(6):1902–1911, 2020.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.

[PD16] Joseph Poon and Thaddeus Dryja. The bitcoin lightnight network: Scalable off-chain instant payments. Technical report, Lightning Network, 2016.

[Ped91] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *IEEE S&P*, May 2013.

[Pie19] Krzysztof Pietrzak. Simple verifiable delay functions. In *ITCS*, volume 124 of *LIPIcs*, pages 60:1–60:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[RS20] Lior Rotem and Gil Segev. Generically speeding-up repeated squaring is equivalent to factoring: sharp thresholds for all generic-ring delay functions. In *CRYPTO*, 2020.

[RSW96] Ronald L. Rivest, Adi Shamir, and David A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, Massachusetts Institute of Technology, 1996.

[RTK90] Michael H Rothkopf, Thomas J Teisberg, and Edward P Kahn. Why are vickrey auctions rare? *Journal of Political Economy*, 98(1):94–109, 1990.

[Set20] Srinath Setty. Spartan: Efficient and general-purpose zksnarks without trusted setup. In *CRYPTO (3)*, volume 12172 of *Lecture Notes in Computer Science*, pages 704–737. Springer, 2020.

[SJH$^+$21] Philipp Schindler, Aljosha Judmayer, Markus Hittmeir, Nicholas Stifter, and Edgar R. Weippl. Randrunner: Distributed randomness from trapdoor vdfs with strong uniqueness. In *NDSS*. The Internet Society, 2021.

[SJK$^+$17] Ewa Syta, Philipp Jovanovic, Eleftherios Kokoris-Kogias, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Michael J. Fischer, and Bryan Ford. Scalable bias-resistant distributed randomness. In *IEEE Symposium on Security and Privacy*, pages 444–460. IEEE Computer Society, 2017.

[SJSW20] Philipp Schindler, Aljosha Judmayer, Nicholas Stifter, and Edgar R. Weippl. Hydrand: Efficient continuous distributed randomness. In *IEEE Symposium on Security and Privacy*, pages 73–89. IEEE, 2020.

[SM21] Jan Christoph Schlegel and Akaki Mamageishvili. On-chain auctions with deposits. *arXiv preprint arXiv:2103.16681*, 2021.

[sot21] Sotheby's to Announce Live Bidding Increments in Ether (ETH) Cryptocurrency for Banksy's 'Trolley Hunters' and 'Love Is In The Air'. Sotheby's press release, Nov 2021.

[SSV21] Cosimo Sguanci, Roberto Spatafora, and Andrea Mario Vergani. Layer 2 blockchain scaling: a survey. *CoRR*, abs/2107.10881, 2021.

[Sta96] Markus Stadler. Publicly verifiable secret sharing. In *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 190–199. Springer, 1996.

[SVS$^+$21] Gaurav Sharma, Denis Verstraeten, Vishal Saraswat, Jean-Michel Dricot, and Olivier Markowitch. Anonymous Sealed-Bid Auction on Ethereum. *Electronics*, 10(19):2340, 2021.

[TBM$^+$20] Sri Aravinda Krishnan Thyagarajan, Adithya Bhat, Giulio Malavolta, Nico Döttling, Aniket Kate, and Dominique Schröder. Verifiable timed signatures made practical. In *CCS*, pages 1733–1750. ACM, 2020.

[TCLM21] Sri Aravinda Krishnan Thyagarajan, Guilhem Castagnos, Fabien Laguillaumie, and Giulio Malavolta. Efficient CCA timed commitments in class groups. In *CCS*, pages 2663–2684. ACM, 2021.

[TGB$^+$21] Sri Aravinda Krishnan Thyagarajan, Tiantian Gong, Adithya Bhat, Aniket Kate, and Dominique Schröder. Opensquare: Decentralized repeated modular squaring service. In *CCS*, pages 3447–3464. ACM, 2021.

[Tha20] Justin Thaler. Proofs, arguments, and zero knowledge. https://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.html, 2020.

[TKFJ22] Weizhao Tang, Lucianna Kiffer, Giulia Fanti, and Ari Juels. Strategic latency reduction in blockchain peer-to-peer networks. *CoRR*, abs/2205.06837, 2022.

[TYME21] Itay Tsabary, Matan Yechieli, Alex Manuskin, and Ittay Eyal. MAD-HTLC: because HTLC is crazy-cheap to attack. In *IEEE Symposium on Security and Privacy*, pages 1230–1248. IEEE, 2021.

[Vic61] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance*, 16(1):8–37, 1961.

[WBJP20] Riad S. Wahby, Dan Boneh, Christopher Jeffrey, and Joseph Poon. An airdrop that preserves recipient privacy. In *Financial Cryptography*, volume 12059 of *Lecture Notes in Computer Science*, pages 444–463. Springer, 2020.

[Wes19] Benjamin Wesolowski. Efficient verifiable delay functions. In *EUROCRYPT (3)*, volume 11478 of *Lecture Notes in Computer Science*, pages 379–407. Springer, 2019.

[WGH$^+$] Barry Whitehat, Alex Gluchowski, HarryR, Yondon Fu, and Philippe Castonguay. Roll up / roll back snark. https://ethresear.ch/t/roll-up-roll-back-snark-side-chain-17000-tps/.

[XW19] Jie Xiong and Qi Wang. Anonymous Auction Protocol Based on Time-Released Encryption atop Consortium Blockchain. *arXiv preprint arXiv:1903.03285*, 2019.

[zca] ZCash. https://z.cash/.

[ZMEF22] Haoqian Zhang, Louis-Henri Merino, Vero Estrada-Galiñanes, and Bryan Ford. F3B: A low-latency commit-and-reveal architecture to mitigate blockchain front-running. *CoRR*, abs/2205.08529, 2022.

A SECURITY DEFINITIONS AND PROOFS FOR TTD

In Appendix A.1, we formally define the security properties for timed commitments. The properties for non-timed commitments are the same while ignoring ForceOpen and the time bound t. Then, in Appendix A.2, we provide proofs for our main timed commitment construction TTD given in Figure 1. In Appendix A.3, we discuss the functional non-malleability notion that we use in this work in greater detail, including how it is used in context of decentralized auctions.

A.1 Timed Commitment Security Definitions

We say that a timed commitment, given by the algorithms (Setup, Comm, ForceOpen, VerOpen), is a secure timed commitment if it satisfies the following properties. The definition below closely follows the definition of [FKPS21] extended to our setting.

- *Full Correctness*: For every $\lambda, t \in \mathbb{N}$, $pp \in \text{Supp}\,(\mathsf{Setup}(\lambda, t))$, and string $com \in \{0,1\}^*$, the following hold:
  - If $(com, \cdot) \in \text{Supp}\,(\mathsf{Comm}^{pp}(b))$ for some $b \in \{0,1\}^*$, then $\mathsf{ForceOpen}^{pp}(com) = (b, \cdot)$.
  - If $(com, \cdot) \notin \text{Supp}\,(\mathsf{Comm}^{pp}(b))$ for any $b \in \{0,1\}^*$, then $\mathsf{ForceOpen}^{pp}(com) = (\perp, \cdot)$.
- *Completeness*: For every $\lambda, t \in \mathbb{N}$, $pp \in \text{Supp}\,(\mathsf{Setup}(\lambda, t))$, and string $com \in \{0,1\}^*$, the following hold:
  - If $(com, \pi_{\mathsf{Open}}) \in \text{Supp}\,(\mathsf{Comm}^{pp}(b))$ for any $b \in \{0,1\}^*$, then $\mathsf{VerOpen}^{pp}(com, b, \pi_{\mathsf{Open}}) = 1$.
  - If $(b, \pi_{\mathsf{Open}}) \leftarrow \mathsf{ForceOpen}^{pp}(com)$, then $\mathsf{VerOpen}^{pp}(com, b, \pi_{\mathsf{Open}}) = 1$.
  - If $\mathsf{VerOpen}^{pp}(com, b, \pi_{\mathsf{Open}}) = 1$ for any $b, \pi_{\mathsf{Open}} \in \{0,1\}^*$, then $(com, \cdot) \in \text{Supp}\,(\mathsf{Comm}^{pp}(b))$.
- *Efficiency*: For every polynomial q_1, there exists a polynomial q_2 such that for every $\lambda, t \in \mathbb{N}$, $pp \in \text{Supp}\,(\mathsf{Setup}(\lambda, t))$, and for inputs bounded by length $q_1(\lambda)$, Setup, Comm^{pp}, and $\mathsf{VerOpen}^{pp}$ run in time at most $q_2(\lambda)$, and $\mathsf{ForceOpen}^{pp}$ runs in time at most $t \cdot q_2(\lambda)$.
- *Binding*: For every non-uniform PPT A, there exists a negligible function negl such that for all $\lambda, t \in \mathbb{N}$,

$$\Pr\left[\begin{array}{l} pp \leftarrow\!\!\$\ \mathsf{Setup}(\lambda, t) \\ (com, (b_0, \pi_{\mathsf{Open},0}), (b_1, \pi_{\mathsf{Open},1})) \leftarrow A(pp) \end{array} : \begin{array}{l} \mathsf{VerOpen}^{pp}(com, b_0, \pi_{\mathsf{Open},0}) = 1 \\ \wedge\ \mathsf{VerOpen}^{pp}(com, b_1, \pi_{\mathsf{Open},1}) = 1 \\ \wedge\ b_0 \neq b_1 \end{array}\right] \leq \text{negl}(\lambda).$$

  Note that b_0 or b_1 output by the adversary A may be equal to $\perp$.

Defining non-malleability. Let L be a bound on the length of a bid, and let $\mathcal{F}$ be a class of functions that takes inputs of the form $(\perp \cup \{0,1\}^L)^*$. We next define the notion of functional non-malleability for the class of functions $\mathcal{F}$, extending the definition of Freitag et al. [FKPS21] to our setting. To define functional non-malleability, we introduce the notion of a meddler-in-the-middle (MIM) adversary. We consider 1-many concurrent non-malleability, which is known to imply many-many concurrent non-malleability [FKPS21].

For any $\lambda, t \in \mathbb{N}$, $pp \in \text{Supp}\,(\mathsf{Setup}(\lambda, t))$, MIM adversary A, and bid $b \in \{0,1\}^L$, we define the experiment $\mathsf{mim}_A(pp, b)$ as follows. An n-MIM adversary A takes as input pp and com as given by $\mathsf{Comm}^{pp}(b)$. A then outputs a sequence of n values com_i for $i \in [n]$. For each $i \in [n]$, we set $\widehat{b}_i$ equal to $(\widehat{b}_i, \cdot) \leftarrow \mathsf{ForceOpen}^{pp}(com_i)$ if $com_i \neq com$. Otherwise, we set $\widehat{b}_i = \perp$. The output of the experiment $\mathsf{mim}_A(pp, b)$ is $\widehat{b}_1, \ldots, \widehat{b}_n$.

We say that (Setup, Comm, ForceOpen, VerOpen) satisfies (concurrent) functional non-malleability with respect to the class of functions $\mathcal{F}$ if the following holds:

- *Functional Non-Malleability*: Let L be a bound on the bid length. For every function $f \in \mathcal{F}$ and polynomial n, there exists a polynomial α such that for all polynomial time bounds $T \geq \alpha$, n-MIM adversary A running in parallel time less than $T(\lambda)/\alpha(\lambda)$, and polynomial-time distinguisher D, there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$, $t = T(\lambda)$, bids $b_0 \neq b_1 \in \{0,1\}^L$, it holds that

$$\Pr\left[\begin{array}{l} pp \leftarrow\!\!\$\ \mathsf{Setup}(\lambda, t) \\ i \leftarrow\!\!\$\ \{0,1\} \\ b_0, b_1 \leftarrow A(pp) \\ \widehat{b}_1, \ldots, \widehat{b}_{n(\lambda)} \leftarrow \mathsf{mim}_A(pp, b_i) \\ i' \leftarrow D(pp, \widehat{b}_1, \ldots, \widehat{b}_{n(\lambda)}) \end{array} : i' = i\right] \leq 1/2 + \text{negl}(\lambda).$$

Random oracle model formalism. We formalize timed commitments in the random oracle model by giving all algorithms oracle access to a hash function H modeled as a random function. For each security parameter $\lambda \in \mathbb{N}$, we assume that the hash function H has fixed input and output length, bounded by a polynomial in λ. Furthermore, we assume for simplicity that any two distinct inputs have a uniformly and independently sampled output (to deal with concrete issues surrounding padding inputs to a fixed length). We require that the correctness properties hold for any choice of hash function H, and we require that the security properties hold where the probability is additionally over a uniformly sampled hash function H.

A.2 Security Proofs for TTD

Before providing proofs, we first formally state the assumptions we need for our main claims.

Assumptions. We recall the repeated squaring assumption, based on the time-lock puzzle proposal of [RSW96], which we refer to as the RSW assumption. Informally, the assumption states that no adversary, on input a random RSA group element x, can compute $y = x^{2^t} \pmod N$ while running in parallel time less than t. Our formal definition is modeled off of the definition of time-lock puzzles given in [FKPS21].

Definition A.1 (The RSW Assumption). Let B be a function of the security parameter. We say that the RSW assumption holds against $B(\lambda)$-time attackers if there is a positive polynomial α such that for all polynomial time bounds $T \geq \alpha$, non-uniform $B(\lambda)$-time pre-processing algorithms A_1, and non-uniform online adversaries A_2 running in parallel time less than $T(\lambda)/\alpha(\lambda)$ and total time less than $B(\lambda)$, there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$, $t = T(\lambda)$,

$$\Pr\left[\begin{array}{l} (\cdot,\cdot,N,\mathbb{G},g,\cdot) \leftarrow\!\!\$\ \mathsf{RSAGGen}(\lambda) \\ \rho \leftarrow A_1(\lambda, N) \\ z \leftarrow A_2(g,\rho) \end{array} : z = g^{2^t} \pmod N\right] \leq \frac{\mathrm{negl}(\lambda)}{B(\lambda)}.$$

Next, we define the properties we need for our underlying encryption scheme. In addition to semantic security, we require a simulatable variant of IND-CCA security. First, we recall that definition of semantic security as follows.

Definition A.2. We say that an encryption scheme $(\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ is semantically secure if for every pair of non-uniform PPT algorithms $A = (A_1, A_2)$, there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$, it holds that

$$\Pr\left[\begin{array}{l} k \leftarrow \mathsf{Keygen}(\lambda) \\ (m_0, m_1, \rho) \leftarrow A_1 \\ i \leftarrow\!\!\$\ \{0,1\} \\ ct \leftarrow \mathsf{Enc}(k, m_i) \\ i' \leftarrow A_2(ct, \rho) \end{array} : \begin{array}{l} i' = i \\ |m_0| = |m_1| \end{array}\right] \leq \mathrm{negl}(\lambda).$$

Informally, we rely on the fact that no attacker, on input a ciphertext for a random key, can produce any other valid ciphertext. We refer to such a scheme as CCA-SIM-1-secure.

Definition A.3. We say that an encryption scheme $(\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ is CCA-SIM-1-secure if it is a semantically secure encryption scheme and the following holds. For every non-uniform PPT A, there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$, $m \in \{0,1\}^*$, it holds that

$$\Pr\left[\begin{array}{l} k \leftarrow \mathsf{Keygen}(\lambda) \\ ct \leftarrow \mathsf{Enc}(k, m) \\ ct' \leftarrow A(ct) \end{array} : \begin{array}{l} \mathsf{Dec}(k, ct') \neq \bot \\ ct \neq ct' \end{array}\right] \leq \mathrm{negl}(\lambda).$$

We note that the standard "encrypt-then-MAC" construction of a symmetric-key CCA-secure encryption scheme satisfies this stronger property. However, it is not immediately implied by plain IND-CCA security (as there may exist fixed strings that are valid ciphertexts under any key).

Security proofs. We provide full proofs for the security properties binding and functional non-malleability. However, the proofs for full correctness, completeness, and efficiency are straightforward, so we omit the full details. We note that full correctness follows immediately from the completeness of the underlying commitment scheme and the correctness of the encryption scheme. Completeness also follows from completeness of the underlying commitment scheme and correctness of the encryption scheme, and additionally relies on the completeness of the proof of exponentiation. Efficiency follows immediately from inspection of the algorithms.

We proceed to prove binding of our timed commitment scheme.

Lemma A.4. *Assuming correctness of the encryption scheme* CCA *and soundness of the proof exponentiation* PoE*,* TTD *satisfies binding.*

Proof. Suppose there exists a polynomial q and a non-uniform PPT A that on input $pp \leftarrow\!\!\$\ \mathsf{Setup}(\lambda, t)$ outputs $(com, (b_0, \pi_{\mathsf{Open},0}), (b_1, \pi_{\mathsf{Open},1}))$ and violates binding with probability at least $1/q(\lambda)$.

Let $com = (com_{\mathsf{C}}, (\hat{h}, ct))$. Consider the event where the binding experiment outputs 1. For $i \in \{0,1\}$, let b_i' be the bid value computed via decryption by $\mathsf{VerOpen}(com, b_i, \pi_{\mathsf{Open},i})$. As $\mathsf{VerOpen}$ outputs 1, it follows that $b_i' = b_i$ and hence $b_0' \neq b_1'$. If both b_0 and b_1 are non-$\bot$, binding follows immediately from binding of the underlying commitment scheme, but we still need to cover the case where one of the bids is $\bot$ and the other is non-$\bot$.

For $i \in \{0,1\}$, let y_i be the element used to generate the corresponding encryption key $k_i = \mathsf{H}(y_i, pp)$ during $\mathsf{VerOpen}(com, b_i, \pi_{\mathsf{Open},i})$. Recall that if $\mathsf{mode} = \texttt{force}$, y_i is provided (with a proof of exponentiation) in $\pi_{\mathsf{Open},i}$, and if $\mathsf{mode} = \texttt{committer}$, y_i is computed as y^{α_i} for a randomization exponent α_i provided in $\pi_{\mathsf{Open},i}$.

First, suppose that $y_0 = y_1$ in the event that the binding experiment outputs 1. This implies that the generated keys k_0 and k_1 are also equal, so it holds that $b_0 = b_1$ by correctness of the encryption scheme, in contradiction. To analyze the case where $y_0 \neq y_1$, we split the event that the binding experiment outputs 1 into three sub-cases: (1) mode = `committer` for both $\pi_{\mathsf{Open},0}$ and $\pi_{\mathsf{Open},1}$, (2) mode = `committer` for $\pi_{\mathsf{Open},0}$ and mode = `force` for $\pi_{\mathsf{Open},1}$ (or vice versa), and (3) mode = `force` for both $\pi_{\mathsf{Open},0}$ and $\pi_{\mathsf{Open},1}$. Thus, one of (1), (2), or (3) must occur with probability at least $1/(3q(\lambda))$, which we show violates our assumptions.

In case (1), we have that mode = `committer` for $\pi_{\mathsf{Open},0}$, so for some α_0 provided in $\pi_{\mathsf{Open},0}$, we have that $y_0 = z^{\alpha_0} = (h^{2^t})^{\alpha_0} = (h^{\alpha_0})^{2^t} = \hat{h}^{2^t} \pmod N$. However, since mode = `committer` for $\pi_{\mathsf{Open},1}$, it also holds that $y_1 = \hat{h}^{2^t} \pmod N$, so $y_0 = y_1$, in contradiction.

In case (2), mode = `force` for $\pi_{\mathsf{Open},1}$, so A provides an accepting proof of exponentiation that y_1 is equal to $\hat{h}^{2^t} \pmod N$. However, $\hat{h}^{2^t} = y_0$ by our analysis of case (1). So if $y_0 \neq y_1$, this violates soundness of the proof of exponentiation. As A is a non-uniform PPT algorithm, this implies that case (2) cannot occur with $1/(3q(\lambda))$ probability.

For case (3), mode = `force` for both $\pi_{\mathsf{Open},0}$ and $\pi_{\mathsf{Open},1}$, and recall that $y_0 \neq y_1$. This means that A provides an accepting proof of exponentiation that $\hat{h}^{2^t}$ is equal to both y_0 and y_1, but at least one of these values must not be correct. Thus, this violates soundness of the proof of exponentiation and cannot occur with $1/(3q(\lambda))$ probability. □

We next prove that TTD satisfies functional non-malleability for the class of functions $\mathcal{F}_\ell$ that can be computed "in low depth" and have output length bounded by $\ell(\lambda)$. At a high level, we require that any function $f \in \mathcal{F}_\ell$ can be computed in parallel time less than the time parameter t used in our construction, for any (polynomial) number of participants n. We formalize this requirement to capture functions that don't depend on the time bound t following [FKPS21]. Specifically, let $f \in \mathcal{F}_\ell$ be a function. We require that there exists a polynomial d such that for any polynomial n representing the number of inputs to f, f can be computed in parallel time less than $d(\lambda, \log n(\lambda))$. This ensures that for large enough (yet polynomial) time bounds $T(\lambda)$, f can be computed in parallel time less than $T(\lambda)$ for any $n(\lambda) \leq 2^\lambda$.

Lemma A.5. *Let ℓ be a function of the security parameter. Assuming the underlying encryption scheme* CCA *is* CCA-SIM-1 *secure, the RSW assumption holds against $2^{\ell(\lambda)} \cdot \mathrm{poly}(\lambda)$-time attackers, and* C *satisfies concurrent functional non-malleability for $\mathcal{F}_\ell$,* TTD *satisfies concurrent functional non-malleability for $\mathcal{F}_\ell$.*

Proof. For any MIM adversary A, distinguisher D, and function $f \in \mathcal{F}$, let G_i be the game representing the output of D in the functional non-malleability security game when the bit i is chosen. By way of contradiction, suppose there is a valid MIM adversary A and a polynomial-time distinguisher D that computes $i' = i$ on game G_i for a random $i \leftarrow\$ \{0,1\}$ with better than $1/2 + 1/q(\lambda)$ probability for some polynomial q. We define a sequence of hybrid games as follows for each $i \in \{0,1\}$ as follows:

- Hybrid $H_{i,0}$: This game is identical to G_i.
- Hybrid $H_{i,1}$: This game is the same as hybrid $H_{i,0}$, except instead of determining the bids $\widehat{b}_1, \ldots, \widehat{b}_n$ using $\mathsf{ForceOpen}((com_i, (\hat{h}_i, ct_i)))$, we use the function OpenSim, defined as follows. $\textsc{OpenSim}((com_i, (\hat{h}_i, ct_i)))$ checks if the adversary has queried $k_i = H(\hat{z}_i, pp)$ such that $\hat{z}_i = (\hat{h}_i)^{2^t} \pmod N$.
  If so, it outputs $\tilde{b}_i$ if $(\tilde{b}_i, \pi_{\mathsf{Open},i}) = \mathsf{CCA.Dec}(k_i, ct_i)$ and $\mathsf{C.VerOpen}(com_i, \tilde{b}_i, \pi_{\mathsf{Open},i}) = 1$. Otherwise, it outputs $\perp$.
- Hybrid $H_{i,2}$: In this hybrid, $\mathsf{TTD.Comm}$ computes the key $k = H(r, pp)$ for a randomly chosen group element r.
- Hybrid $H_{i,3}$: Same as $H_{i,2}$, except $\mathsf{TTD.Comm}$ computes $ct \leftarrow \mathsf{CCA.Enc}(k, \vec{0})$ such that $|\vec{0}| = |(b, \pi_{\mathsf{Open}})|$.
- Hybrid $H_{i,4}$: This hybrid is the same as $H_{i,3}$, except $\mathsf{TTD.Comm}$ computes $com_{\mathsf{C}} \leftarrow \mathsf{C.Comm}(0^L)$.

Let $p_{i,j}$ be the probability that D outputs 1 in hybrid $H_{i,j}$. As $H_{0,0}$, $H_{1,0}$ correspond to the game when b_0 or b_1 are chosen, respectively, this means that $p_{1,0}/2 + (1 - p_{0,0})/2 \geq 1/2 + 1/q(\lambda)$, which implies that $|p_{0,0}/2 - p_{1,0}/2| \geq 1/q(\lambda)$. As $H_{0,4}$ and $H_{1,4}$ are independent of b_0 and b_1, respectively, it follows that the games are identical so $p_{0,4} = p_{1,4}$. So, it must be the case that $|p_{i,j}/2 - p_{i,j+1}/2|$ for some $i \in \{0,1\}$ and $j \in \{0,1,2,3\}$ must be at least $1/(8q(\lambda))$. In the following claims, we show that this must contradict our assumptions. Without loss of generality, we prove the claims for $i = 0$.

Claim 1. *Assuming* CCA *is* CCA-SIM-1 *secure,* $|p_{0,0}/2 - p_{0,1}/2| \leq 1/(8q(\lambda))$.

Proof. For every $\lambda \in \mathbb{N}$, we consider a sequence of $n + 1$ additional hybrids $J_0, \ldots, J_n$, where $J_0 = H_{0,0}$ and $J_n = H_{0,1}$. In hybrid J_k, the first $n - k$ bids $\widehat{b}_j$ corresponding to commitments output by A are computed using ForceOpen, and the remaining k bids $\widehat{b}_j$ are computed using OpenSim. Thus, J_k and J_{k+1} differ solely in the way that the $k + 1$st bid $\widehat{b}_{k+1}$ is computed. We show that if, for any $k \in [0, n-1]$, A distinguishes J_k from J_{k+1} with greater than $1/(8 \cdot n \cdot q)$ probability, we can use A to break the CCA-SIM-1 security of the underlying encryption scheme CCA.

Consider any $j \in [0, n-1]$. Let $(com_{j+1}, (\hat{h}_{j+1}, ct_{j+1}))$ be the $j + 1$st commitment output by A. The output of the game is identically distributed, unless ForceOpen and OpenSim compute different values for $\widehat{b}_{j+1}$, in which case we make no assumptions on the output of the game.

Suppose that $\textsc{OpenSim}$ outputs a valid $\tilde{b}_{j+1} \neq \perp$. By definition of $\textsc{OpenSim}$, this implies that $(\tilde{b}_{j+1}, \pi_{\mathsf{Open},j+1}) = \mathsf{CCA.Dec}(k, ct_{j+1})$ where $\hat{z}_{j+1} = (\hat{h}_{k+1})^{2^t}$, $k = \mathsf{H}(\hat{z}_{j+1}, pp)$, and $\mathsf{C.VerOpen}(com_{j+1}, \tilde{b}_{j+1}, \pi_{\mathsf{Open},j+1}) = 1$. But this means that $(com_{j+1}, (\hat{h}_{j+1}, ct_{j+1}))$ is a valid commitment for $\tilde{b}_{j+1}$, so $\mathsf{ForceOpen}$ outputs $\widehat{b}_{j+1} = \tilde{b}_{j+1}$. Thus, the output of the game can only differ between J_j and J_{j+1} if $\textsc{OpenSim}$ outputs $\perp$ and $\mathsf{ForceOpen}$ outputs a value $\widehat{b} \neq \perp$.

Suppose $\textsc{OpenSim}$ outputs $\perp$ on the $j + 1$st commitment even though $(com_{j+1}, (\hat{h}_{j+1}, ct_{j+1}))$ is a valid commitment for some value $\widehat{b}_{j+1} \neq \perp$. This implies that ct_{j+1} is a valid encryption under the key $k = \mathsf{H}(\hat{z}_{j+1}, pp)$, but A has not queried H on $(\hat{z}_{j+1}, pp)$. Thus, k is a uniformly random key. At most, A has received a single ciphertext $ct = \mathsf{CCA.Enc}(k, (b_0, \pi_{\mathsf{Open},0}))$ under this key (or has not received such a ciphertext if the $k + 1$st query happens before the challenge), and outputs a distinct valid ciphertext under k. By CCA-SIM-1 security, this cannot happen with probability at least $1/(8 \cdot n \cdot q)$, in contradiction. □

Claim 2. *Assuming RSW assumption holds for $2^{\ell(\lambda)} \cdot \text{poly}(\lambda)$-time attackers, $|p_{0,1}/2 - p_{0,2}/2| \leq 1/(8q(\lambda))$.*

Proof. Using the MIM adversary A, the function $f \in \mathcal{F}_\ell$, and distinguisher D, we construct an adversary $B = (B_1, B_2)$ that wins the RSW game with sufficient advantage. Furthermore, B will run in total time at most $2^{\ell(\lambda)} \cdot \text{poly}(\lambda)$ and B_2 will run in low parallel time.

The pre-processing adversary B_1 first receives as input the group description N. It samples the remainder of the public parameters pp for $\mathsf{TTD.Setup}(\lambda, t)$, including a random group element h. It then computes $z = h^{2^t} \pmod N$ by computing t squarings. B_1 next computes com_{C} and ct for $\mathsf{TTD.Comm}^{pp}(b_0)$, using $k = \mathsf{H}(r, pp)$ for a random group element r. Next, B_1 runs D on all possible $\ell(\lambda)$-bit inputs, querying H up to $2^{\ell(\lambda)} \cdot \text{poly}(\lambda)$ times, and lets $\vec{y}$ denote the sequence of values y such that D queried $\mathsf{H}(y, pp)$. B_1 outputs its state $\rho = (pp, com_{\mathsf{C}}, ct, \vec{y})$.

Next, the online adversary B_2 receives a random group element x and the state ρ from B_1. B_2 runs $A(com_{\mathsf{C}}, (x, ct))$ and gets the output $(com_i, (\hat{h}_i, ct_i))$ for all $i \in [n]$. B_2 computes $\widehat{b}_i \leftarrow \textsc{OpenSim}((com_i, (\hat{h}_i, ct_i)))$ for each output in parallel, and then B_2 runs $f(\widehat{b}_1, \ldots, \widehat{b}_n)$. Let $\vec{y}'$ be the sequence of values y' such that A or f queried $\mathsf{H}(y', pp)$. B_2 chooses a random value $\tilde{y}$ from $\vec{y}$ or $\vec{y}'$ and outputs $\tilde{y}$.

We first analyze the running time of B. B_1 takes time $t \cdot \text{poly}(\lambda)$ to compute z and takes time $2^{\ell(\lambda)} \cdot \text{poly}(\lambda)$ to compute $\vec{y}$, so for polynomial time bounds $T(\lambda)$, B_1 runs in time $2^{\ell(\lambda)} \cdot \text{poly}(\lambda)$. The online adversary B_2 runs A, invokes $\textsc{OpenSim}$ on each output in parallel, and then runs $f(\widehat{b}_1, \ldots, \widehat{b}_n)$. Let t_A be the parallel running time of A. All $\textsc{OpenSim}$ evaluations can be computed in parallel time $p_1(\lambda)$, for a polynomial p_1 independent of the time parameter t. By assumption on $\mathcal{F}_\ell$, f can be computed in time $d(\lambda, \log n(\lambda))$. For $n(\lambda) \leq 2^\lambda$, this implies that B_2 runs in time $t_A + p_2(\lambda) \leq T(\lambda)/\alpha(\lambda) + p_2(\lambda)$ for a fixed polynomial p_2 independent of the time parameter t. For the running time of A, recall that for every positive polynomial α, there exists a polynomial $T \geq \alpha$ such that A succeeds and runs in parallel time at most T/α. To show that B violates our assumption on the running time, let α' be an arbitrary positive polynomial. We invoke the assumed adversary A for the value $\alpha = 2\alpha' \cdot p_2$, so $T \geq 2\alpha' \cdot p_2$. Then, by our analysis above, B runs in parallel time at most $T/\alpha + p_2 = (T + \alpha \cdot p_2)/\alpha \leq 2T/\alpha \leq T/\alpha'$. Thus, B has the appropriate parallel running time, and it remains to argue B violates the RSW assumption the required probability.

Consider a hypothetical experiment where B_2 computes the key k for the challenge using $y = x^{2^t} \pmod N$. This corresponds to hybrid $H_{0,1}$, whereas B_2's real behavior corresponds to hybrid $H_{0,2}$. However, the output of hybrids $H_{0,1}$ and $H_{0,2}$ are identically distributed unless A, f, or D query the random oracle H on either (1) the input (r, pp) where r was the randomly chosen group element in the experiment or (2) (y, pp) for $y = x^{2^t} \pmod N$. This is because in both cases, the challenge key k generated is uniformly random and identically distributed in each hybrid. Let $m \leq 2^{\ell(\lambda)} \cdot \text{poly}(\lambda)$ be the number of queries in $\vec{y}$ or $\vec{y}'$. Case (1) happens with at most $m/2^\lambda$ probability since r is a random, independently chosen group element, and there are at least 2^λ group elements in $\mathbb{G}$. In case (2), B succeeds in the RSW game with probability at least $1/m$. Therefore, B succeeds with probability at least $(|p_{0,1}/2 - p_{0,2}/2| - m/2^\lambda)/m > |p_{0,1}/2 - p_{0,2}/2|/m - 1/2^\lambda$. If $|p_{0,1}/2 - p_{0,2}/2| > 1/8q$, this implies the existence of a polynomial q' such that B succeeds with probability at least $1/(q' \cdot 2^{\ell(\lambda)})$, contradicting our RSW assumption against $2^{\ell(\lambda)} \cdot \text{poly}(\lambda)$-time attackers, as required. □

Claim 3. *Assuming CCA is semantically secure, $|p_{0,2}/2 - p_{0,3}/2| \leq 1/(8q(\lambda))$.*

Proof. Using the MIM adversary A, function f, and distinguisher D that distinguish hybrids $H_{0,2}$ and $H_{0,3}$, we construct an adversary $B = (B_1, B_2)$ that breaks semantic security as follows. B_1 first samples public parameters pp for TTD. It then computes $(com_{\mathsf{C}}, \pi_{\mathsf{Open},C}) \leftarrow \mathsf{C.Comm}(b_0)$. B_1 sends $m_0 = (b_0, \pi_{\mathsf{Open},C})$ and $m_1 = \vec{0}$ such that $|\vec{0}| = |(b_0, \pi_{\mathsf{Open},C})|$ to the semantic security challenger. B_2 receives ct corresponding to an encryption of m_i for either $i = 0$ or $i = 1$. It then computes the commitment $com = (com_{\mathsf{C}}, (\hat{h}, ct))$ where $\hat{h} \leftarrow h^\alpha \pmod N$ for $\alpha \leftarrow\$ [2^{2\lambda}]$. B_2 runs $A(pp, com)$ and gets commitments $(com_i, (\hat{h}_i, ct_i))$ for all $i \in [n]$. B computes $\widehat{b}_i \leftarrow \textsc{OpenSim}((com_i, (\hat{h}_i, ct_i)))$, and outputs $i' = D(f(\widehat{b}_1, \ldots, \widehat{b}_n))$.

B clearly runs in polynomial-time as the time bound $t = T(\lambda)$ is polynomially bounded. For correctness, note that whenever the semantic security challenger $i = 0$, ct corresponds to the message $(b_0, \pi_{\mathsf{Open,C}})$, so the output of the game is identical to hybrid $H_{0,2}$ since the key k chosen by the semantic security challenger is uniformly random. Similarly, whenever $i = 1$ is chosen, the output of the game is identical to hybrid $H_{0,3}$. Thus, B's distinguishing advantage is $|p_{0,2}/2 - p_{0,3}/2|$, which violates semantic security of the underlying encryption scheme if $|p_{0,2}/2 - p_{0,3}/2| > 1/8q$. □

Claim 4. *Assuming* C *satisfies functional non-malleability for* $\mathcal{F}_\ell$, $|p_{0,3}/2 - p_{0,4}/2| \leq 1/(8q(\lambda))$.

Proof. Given the MIM adversary A, we construct a MIM adversary B for the functional non-malleability game for the underlying commitment scheme C for the function f on messages $m_0 = b_0$ or $m_1 = 0^L$ as follows. B receives as input public parameters pp_{C} and a commitment $com_{\mathsf{C}} \leftarrow \mathsf{C.Comm}^{pp_{\mathsf{C}}}(m_i)$ for some $i \leftarrow\!\!\$ \{0,1\}$. It then samples the remainder of the public parameters for TTD as well as a value $\hat{h}$ and $ct = \mathsf{CCA.Enc}(k, \vec{0})$ as in hybrid $H_{0,3}$. It runs $A(pp, (com_{\mathsf{C}}, (\hat{h}, ct)))$ and gets outputs $(com_i, (\hat{h}_i, ct_i))$ for all $i \in [n]$. B computes $\widehat{b}_i \leftarrow \textsc{OpenSim}((com_i, (\hat{h}_i, ct_i)))$ for each $i \in [n]$ in parallel, and sets $\widehat{com}_i = com_i$ if $\widehat{b}_i \neq \perp$ and $\widehat{com}_i = \perp$ otherwise. B then outputs $(\widehat{com}_1, \ldots, \widehat{com}_n)$.

For the running time analysis, we note that B runs A and then additionally runs in parallel running time $p_1(\lambda)$, for a polynomial p_1 independent of the time parameter t. It follows that B runs in the appropriate parallel running time for a MIM adversary to break the non-malleability game; see the analysis in Claim 2 for full details.

For correctness, we note that if $i = 0$ and B receives a commitment for $m_0 = b_0$, then A's view is identical to hybrid $H_{0,3}$. Similarly, if $i = 1$ and B receives a commitment for $m_1 = 0^L$, then A's view is identical to hybrid $H_{0,4}$. So the outputs $(com_i, (\hat{h}_i, ct_i))$ for $i \in [n]$ are distributed exactly according to $H_{0,3}$ when $i = 0$ and according to $H_{0,4}$ when $i = 1$. Furthermore, B can compute the bids $\widehat{b}_i$ underlying com_i that are fed into f and then the distinguisher D by running $\textsc{OpenSim}$ identically as in hybrids $H_{0,3}$ and $H_{0,4}$. It then sets commitments $\widehat{com}_i$ to open to the same $\widehat{b}_i$ values. It follows that the MIM experiment for B outputs 1 with probability $p_{0,3}$ whenever $i = 0$ and with probability $p_{0,4}$ whenever $i = 1$. Therefore, D distinguishes $m_0 = b_0$ and $m_1 = 0$ for the MIM adversary B with function f with probability $|p_{0,3} - p_{0,4}|$, which is a contradiction if $|p_{0,3}/2 - p_{0,4}/2| > 1/(8q(\lambda))$, as required. ◻

This completes the proof of the lemma. ◻

A.3 Discussion of Functional Non-Malleability

For non-malleability, we follow the treatment of Freitag et al. [FKPS21] by going through functional non-malleability as fully concurrent non-malleability is impossible to achieve in the timed setting. Specifically, for a suitable function f, we show that any meddler-in-the-middle (MIM) attacker running in time less than the time parameter t cannot maul a timed commitment com for a bid b into a sequence of commitments $com_1, \ldots, com_n$ with underlying values $b_1, \ldots, b_n$ such that $f(b_1, \ldots, b_n)$ is related to b. We consider the class of functions $\mathcal{F}_\ell$ that can be computed in low parallel time and have bounded output length ℓ. The reason this bound ℓ is seemingly necessary is to prevent the adversary from encoding the original timed commitment com into the output of $f(b_1, \ldots, b_n)$, as it has full control over $b_1, \ldots, b_n$. For this reason, it is useful to consider functions with output length ℓ which is smaller than the size of the timed commitments com.

We briefly compare with the non-malleability notion proposed in Katz et al. [KLX20]. They consider a CCA-style notion and require that no attacker running in time less than t can distinguish commitments to b_0 from commitments to b_1, even with oracle access to a $\mathsf{ForceOpen}$ oracle (that naively requires at least t time to run). The reason we do not consider this definition is because it doesn't provide a meaningful guarantee of security after t time has elapsed. In fact, it is argued in [FKPS21] that this notion (without oracle access to $\mathsf{ForceOpen}$) is equivalent to functional non-malleability for the class of low depth function $\mathcal{F}_1$ with 1 bit output—essentially, the function f plays the role of the CCA adversary A that outputs a single bit.

In context of our application to decentralized auctions, functional non-malleability provides the following guarantee. Suppose an honest participant submits a timed commitment com for a bid b. A MIM adversary tries to bias the output of the protocol as a function of b in the following way. It computes a series of commitments $com_1, \ldots, com_n$ for underlying bids $b_1, \ldots, b_n$. The function f in functional non-malleability represents the contribution that the values the attacker commits to can affect the final outcome in the protocol. So, for second-priced sealed-bid auctions, f would output the top two bid values from the attacker and the index of the highest bid, which could then be used to determine the output of the protocol. For such an auction, this information is very small, so intuitively, this says that the attacker is very limited in how much it could adversarially influence the result of the auction as a function of another participant's original bid b.

Consider a $(k+1)$-priced auction where the top k bidders receive a copy of the item, and pay the $(k+1)$st price. If k is large enough, a MIM attacker could bid large values to ensure that participants $i_1, i_2, \ldots, i_k$ win the auction, but simultaneously have it be the case that the concatenation of $i_1|\ldots|i_k \in \{0,1\}^*$ as a string write out the bit string for a timed commitment com for a bid b that another participant provided. This means that the attacker forced the output of the protocol to non-trivially depend on one of the bids b from another honest participant, technically violating fairness for the auction! While this counterexample is somewhat contrived—and very likely is not financially in the attacker's interest—it highlights the subtlety and care required to guarantee a meaningful notion of security for decentralized auctions.

In summary, we decide to formalize our non-malleability guarantees using functional non-malleability for the class of length bounded functions, following [FKPS21]. We believe this captures a broad class of auction-types and also provides a framework to understand specific auctions as needed. Furthermore, we note that instead of looking at the outcome of the auction protocol as a whole, we could instead consider the outcome for any particular participant, which is likely to have short representation and therefore cannot be biased based on their submitted bid by our analysis. For this reason, we believe any counterexamples to fairness in natural protocols are likely contrived and also likely not game-theoretically (i.e. financially) realistic.

```
HTC.Setup(λ, t)
$(q_1, q_2, N, \mathbb{G}, g, h) \leftarrow\!\!\$\ \mathsf{RSAGGen}(\lambda)$
$z \leftarrow h^{(2^t \bmod \varphi(N))} \pmod N$
Return $pp = (N, h, z, t)$

HTC.Comm^{pp}(b)
$\alpha \leftarrow\!\!\$\ [2^{2\lambda}]$
$\hat{h} \leftarrow h^{\alpha} \pmod N\ ;\ \hat{z} \leftarrow z^{\alpha} \pmod N$
$k \leftarrow \mathsf{H}(\hat{z}, pp)$
$ct \leftarrow\!\!\$\ \mathsf{CCA.Enc}(k, b)$
Return $(com = (\hat{h}, ct), \pi_{\mathsf{Open}} = (\texttt{committer}, \alpha))$

HTC.ForceOpen^{pp}(com = (ĥ, ct))
$\hat{z} \leftarrow \hat{h}^{2^t} \pmod N$
$\pi_{\mathsf{PoE}} \leftarrow\!\!\$\ \mathsf{PoE.Prove}(N, \hat{h}, \hat{z}, t)$
$k \leftarrow \mathsf{H}(\hat{z}, pp)$
$b \leftarrow \mathsf{CCA.Dec}(k, ct)\ ;\ \pi_{\mathsf{Open}} \leftarrow (\hat{z}, \pi_{\mathsf{PoE}})$
Return $(b, (\texttt{force}, \pi_{\mathsf{Open}}))$

HTC.VerOpen^{pp}(com = (ĥ, ct), b, π_Open = (mode, π))
If $\texttt{mode} = \texttt{force}$
    Parse $\pi = (\hat{z}, \pi_{\mathsf{PoE}})$
    $k \leftarrow H(\hat{z}, pp)\ ;\ b' \leftarrow \mathsf{CCA.Dec}(k, ct)$
    Return $\mathsf{PoE.Ver}((N, \hat{h}, \hat{z}, t), \pi_{\mathsf{PoE}}) = 1 \wedge b \neq b'$
Else if $\texttt{mode} = \texttt{committer}$
    Parse $\pi = \alpha\ ;\ \hat{z} \leftarrow z^{\alpha} \pmod N$
    $k \leftarrow H(\hat{z}, pp)\ ;\ b' \leftarrow \mathsf{CCA.Dec}(k, ct)$
    Return $\hat{h} = h^{\alpha} \pmod N \wedge b = b'$
```

Protocol: Synchronous Auction House with Timed Commitments

Initialization: The auction house is initialized with public parameters for a timed commitment scheme C and delay parameter t. The auction house stores the following for each associated user:

- bal: Account collateral balance of the user.
- $active$: List of active bids in the current epoch for the user.

Auctions proceed in synchronous, sequential epochs, in which each epoch may run many auctions. An epoch consists of the following phases:

Phase 0: Auction registration: An auctioneer may register an auction with the auction house for the upcoming epoch. This registration phase may be pipelined with previous epochs

Phase 1: Bid collection and balance updates

(1) The auction house runs a separate instance of the auction protocol from Figure 2 (with some minor changes) for each registered auction. Call this the beginning of the epoch (time t_0).
(2) Users may place a bid on any auction. The user's collateral is updated to remove the locked opening rewards, $bal \leftarrow bal - rwd_{\mathsf{Open}} - rwd_{\mathsf{Force}}$. If a user's balance is not sufficient for locking rewards, the bid is not accepted. Any accepted bid is added to a user's list of active bids, $active$.
(3) Users may also transfer funds in and out of their account. The account collateral is updated accordingly.
(4) The auction house ends bid collection and locks account collaterals at time $t_0 + t$.

Phase 2: Bid opening and auction results

(1) Bids are opened following the mechanisms of Figure 2 (Phases 2 and 3). However, the results of the auction are not yet determined.
(2) If a user's bid is abandoned in any auction in the non-timed commitment setting, the user's collateral is forfeited: $bal \leftarrow 0$. In the timed commitment setting, the next step does not begin until all bids for all registered auctions are opened.
(3) For each user, the auction house sums the user's opened bids (across auctions). If the sum is greater than the user's collateral bal, all the user's bids for the epoch are marked as invalid, and the user forfeits their rewards had they self-opened.
(4) Auction results are determined by the remaining valid opened bids. Bid amounts that are determined to be transacted as part of the auction results are subtracted from user collaterals.
(5) Lists of active bids for each user are reset in preparation for the next epoch.

Figure 7: (Left) The HTC non-malleable timed commitment protocol parameterized by a proof of exponentiation protocol PoE. The hash function H is modeled as a random oracle. HTC is the timed commitment used within TTD as the timed trapdoor to the C commitment scheme. (Right) Epoch-based auction house protocol to support multiple simultaneous auctions without the use of range proofs.

B EPOCH-BASED SYNCHRONOUS AUCTION HOUSE

We expand on a proposal for an epoch-based synchronous auction house, outlined in Section 3.2. The protocol allows for an efficient auction house that does not incur the overhead of range proofs as in Riggs-RP and Riggs-TC. However, it comes at the disadvantage of requiring auctions to proceed in synchronous epochs with synchronized start and end times.

B.1 HTC: Simple Hash-based Non-Malleable Timed Commitment

Recall the design of the timed commitment TTD used by Riggs-RP and Riggs-TC: The bid is enclosed using a second non-timed commitment scheme C that is amenable to range proofs, and then the opening to the commitment of C is enclosed as a "timed trapdoor". Since the synchronous auction house does not require range proofs, we can simplify this construction, and extract out the "timed trapdoor" component of TTD as a separate timed commitment scheme which we call HTC.

We provide pseudocode for HTC in Figure 7 (left). We provide the following corollaries for the security of HTC that follow from the security proofs of TTD in Appendix A.2.

Corollary B.1. *Assuming correctness of the encryption scheme* CCA *and soundness of the proof exponentiation* PoE, HTC *satisfies binding.*

Corollary B.2. *Let ℓ be a function of the security parameter. Assuming the underlying encryption scheme* CCA *is* CCA-SIM-1 *secure and the RSW assumption holds against $2^{\ell(\lambda)} \cdot \mathrm{poly}(\lambda)$-time attackers,* HTC *satisfies concurrent functional non-malleability for $\mathcal{F}_\ell$.*

B.2 Synchronous Auctions without Range Proofs

As in Riggs-RP, each user is associated with a collateral that is used to back all bids to auctions a user participates in during a particular epoch. Auctions may be registered ahead of time to take place in a particular epoch. Once an epoch begins, auctions proceed in two phases (detailed in Figure 7). In the first phase, bids are collected for ongoing auctions and, in the case of timed commitments, opening rewards are locked up. The bid collection phase ends synchronously for all auctions in the epoch, and after this point, importantly, user collaterals are locked for the entirety of the second phase. This ensures that users cannot adaptively change their collateral during bid opening to validate or invalidate their bids across auctions.

In the second phase, bids are opened (with a self-open phase, followed by a force-open phase in the case of timed commitments). Without DoS protection from timed commitments, if a user fails to self-open a bid, their full collateral is forfeited. In the case of timed commitments, the results of auctions are not determined until all bids from all auctions have been opened. All of a user's opened bids across auctions are summed and compared to their locked collateral. If their collateral covers their bids, then the bids are deemed valid, otherwise *all* the user's bids are discarded. This all-or-none validity property is important to ensure users cannot selectively invalidate bids. Lastly, the remaining valid bids are used to determine the results of each auction, and the process is repeated for the auctions slated for the next epoch.