AStERISK: Auction-based Shared Economy Resolution Markets for Blockchain Platforms

Alberto Sonnino
University College London
alberto.sonnino@ucl.ac.uk

Michał Król
University College London
m.krol@ucl.ac.uk

Argyrios G. Tasiopoulos
University College London
argyrios.tasiopoulos@ucl.ac.uk

Ioannis Psaras
University College London
i.psaras@ucl.ac.uk

Abstract—Recent developments in blockchains and edge computing allow deployment of decentralized shared economy platforms with utility token support, where tokens (in the form of altcoins) secure and reward useful work. However, the majority of the systems being developed, does not provide mechanisms to pair those who offer services (workers) and those who request them (clients), or rely on manual and insecure resolution. AStERISK bridges this gap allowing to perform sealed-bid auctions on blockchains, automatically determine the most optimal price for services, and assign clients to the most suitable workers. AStERISK allows workers to specify a minimal price for their work, and hide submitted bids as well the identity of the bidders without relying on any centralized party at any point. We provide a smart contract implementation of AStERISK and show how to deploy it within the Filecoin network, and perform an initial benchmark on Chainspace, an efficient, new smart contract platform.

I. Introduction

The cloud computing model developed during the last two decades was built on the premise of compute centralization. That is, computing power is geographically and administratively concentrated in compute infrastructures of industrial scale, generally called datacenters. As a result, the majority of users currently rely on clouds for applications such as hosting services, offloading computation, and data storage. However, centralization introduces several drawbacks; cloud computing services act as large central points of failure [1], and make possible for authorities to enforce censorship [2] or violate user privacy [3]. Furthermore, large cloud operators often abuse their market position to effectively force users to trust their service and adapt to operators' rules and prices.

Recent research efforts focus on shared-economy infrastructures, where services are performed by users sharing their resources with each other at the edge of the network [4]. Such infrastructures can provide better quality of service, and reduce the exposure to central points of failures and abuse of power; but require security solutions and reliable incentive mechanisms to compensate the lack of trust between distributed users. Following the recent success of Bitcoin [5] a large amount of cryptocurrency projects implement such a shared economy model and use blockchains to secure their platforms and simplify payments [6], [7], [8], [9], [10], [11].

The vision is to create a decentralized system, where users are incentivized to perform useful work and automatically receive rewards upon completing the assigned tasks.

While multiple projects focus on the crucial task of proving to the network that a service has been successfully completed [12], [13], [14], [15], an equally important task of determining an optimal price for those services has been largely ignored by the community. Furthermore, industrial projects either do not cover this problem or rely on manual user interaction causing multiple scalability and security issues [6], [7], [8], [9]. In this work, we answer to the question of *how blockchains and smart contracts can be leveraged for deriving the price of a service in distributed computing infrastructures and automatically assign service requesters to corresponding workers.*

The providers of the distributed infrastructure are asking for compensation, since admitting user requests imposes operational expenses, while occupying their personal computing as well as storage resources. Therefore, the sufficient participation of infrastructure providers is associated with an incentivisation mechanism that allows them to profit by offering their resources at a price that equals or exceeds, their expenses. That is, the objectives of users and providers are conflicting with each other since the former try to access a service at the lowest possible price while the latter tries to maximise their revenue. Therefore, there is a need for a market mechanism to associate user requests to providers' resources. In AStERISK we introduce the concept of running auctions on the blockchain: in this way, users do not have to trust each other nor any trusted 3rd party, while both parties are achieving their objectives of price minimisation (for clients) and revenue maximisation (for workers). Instead, they inherit security guarantees from the underlying distributed ledger and can be sure that the auction is performed correctly using *Smart Contracts*. However, data submitted to blockchain automatically becomes public and in sealed-bid auctions it is critical to hide all the bids during the auction, as we discuss extensively later. For example, revealing bidders' identity imposes another security threat: revealing that a bidder bought storage space at some specific and publicly known servers makes the servers an easy target for malicious users that want the bidder to lose their data.

In this paper we propose AStERISK— an auction-based shared economy resolution system running on top of distributed ledger. In AStERISK, workers submit their offers to the blockchain together with a minimal price they are willing to accept. Our system hides submitted bids and protects bidders identity using anonymous credentials. In contrast to related work, AStERISK does not rely on a single trusted 3rd party to issue all the credentials, but rather on a set of

multiple, decentralised authorities. Each user can define their own set of trusted parties and is protected from a subset of authorities becoming malicious. Once an auction is finished, AStERISK automatically creates a binding on the blockchain allowing workers to claim money from requester deposits upon submission of a valid proof of useful work. For simplicity, we focus on the case of Filecoin [16], a decentralized storage network, but AStERISK can be easily adapted to work with additional systems implementing a shared economy model.

II. BACKGROUND

a) Blockchain: The blockchain technology [5] implements a distributed, append-only ledger in the form of connected blocks; once information is stored in the blockchain it cannot be removed or altered. Network participants use a consensus protocol to agree on the current state of the ledger, and the system maintains its properties as long as a subset of the participants is honest. Blockchains are used to record cryptocurrencies (*e.g.,* Bitcoin [5], Ethereum [17]) transactions. A common extension consisting of scripting languages enables to include logic as part of the transaction and allows deployment of smart contracts—code submitted to the blockchain and executed by all network participants. AStERISK leverages the high-integrity data structure provided by blockchains, and uses it for accountancy, auditability, and time reference (*e.g.,* time may be defined as the block height of the main chain).

b) Filecoin: At the core of Filecoin lies a Decentralised Storage Market that allows requesters to pay miners, to store data. Specifically, a *Storage Market* acts as an exchange point where clients and miners can advertise their requests and offer resources. The *Network* guarantees that the miners are rewarded by the clients when providing the service. Filecoin uses Proof-of-Replication [15] a scheme that relies on zero-knowledge Succinct Non-interactive ARguments of Knowledge (zk-SNARKs) [18], [19] allowing a Prover to prove possession of data $\mathcal{D}$. To prevent Sybil attacks and allow proving possession of multiple copies of the same file, Proofs-of-Replication are generated over a version of data encrypted under a key ek specified by the requester such that $\mathcal{R}_{ek}^{\mathcal{D}} = \mathsf{Encode}(\mathcal{D}, ek)$. Filecoin requires the Prover to recursively and continuously generate those proofs with randomness r obtained from the most recent block in the blockchain and thus proving possession of the file over time.

c) Anonymous Credentials: Anonymous credentials [20], [21] allow the issuance of credentials to users, and the subsequent unlinkable revelation to a verifier. Users can selectively disclose some of the attributes embedded in the credential or specific functions of these attributes. Most anonymous credentials schemes entrust a single authority with a master credential signature key, allowing a malicious authority to forge any credential. Other schemes do not provide the necessary re-randomization or blind issuing properties necessary to implement general purpose disclosure credentials. To overcome these limitations, AStERISK relies on Coconut [22] which supports distributed threshold issuance of credentials; therefore supporting private attributes, re-randomization, and unlinkable multi-show selective disclosure without relying on a central trusted 3rd party. Coconut is designed for use in the context of blockchains to ensure confidentiality, authenticity and availability even when a subset of credential issuing authorities are malicious or offline; it uses short and computationally efficient credentials that can easily be verified by a smart contract. Colluding authorities can forge Coconut credentials, but cannot break unlinkability and de-anonymize users. Coconut authorities issue credentials without communicating with each other, following a standard key distribution phase; as a result, a large number of authorities may be used to issue credentials without significantly affecting efficiency.

III. DESIGN GOALS

AStERISK associates worker resources to users via the execution of an auction mechanism. Specifically, we consider a system with the following actors:

- **Bidders** - users willing to access a service (*e.g.,* Filecoin users wishing to store some data in the network).
- **Workers** - offer their services to the users (*e.g.,* Filecoin miners wishing to store users' files for a specific price).
- **Authorities** - distributed system responsible to issue credentials allowing users to participate in auctions.

AStERISK assumes that at least a threshold subset of the authorities are honest; all cryptographic operations rely on (or are implied by) the "external Diffie-Helman" (XDH) assumption [23]; and relies on weak synchrony[1] for liveness (but not for safety). These assumptions are inherited from Coconut, and the underlying smart contract platform. Given this threat model, AStERISK achieves the following design goals:

- **Hidden Minimum Price:** Workers can specify a minimum price for which they are willing to perform given actions, and are guaranteed that the winner of the auction bids at least that price. This minimum price is kept private from the bidders until the end of the auction.
- **Bidders Privacy:** Bidders are unlinkable to to their bids. Only the identity of the winner is revealed to the worker (at the end of the auction).
- **Bids Privacy:** Bids are kept private until the end of the auction; bidders submit their bids without knowledge of what other bidders do.
- **Bids Binding:** Bidders cannot change their bids once they are committed.
- **Public Auditability:** Anyone can verify the correct execution of any auction.
- **Fairness:** Bidders are financially penalized if they deviate from the protocol, but cannot be financially penalized if they follow it correctly. Bidders cannot double-spend coins [27], and no authority can steal bidder's funds.
- **Non-Interactivity:** Bidders are not required to interact with each other.
- **Censorship Resistance:** Anyone can act as bidder or worker; the system is resilient to censorship.
- **Distributed Authority:** AStERISK never relies on a single trusted 3rd party.
- **Auction's economic properties:** Involving *i) Efficiency*, in terms of assigning resources to the bidders that value them the most in a computationally feasible way, *ii) Incentive Compatibility* (*Truthfulness*), where bidders and workers benefit by revealing their true valuations, *iii)*

---

[1]Weak synchrony [24] is required by many smart contract platforms [25], [26], and by distributed key generation protocols required by Coconut [22].

Individual Rationality, where both bidders and workers are willing to participate, and *iv) Budget Balance*, where the payments submitted cover workers' compensations.

IV. ASTERISK DESIGN

A. AStERISK Smart Contract

We design AStERISK as a smart contract that extends the tumbler application of Coconut [22] to allow credentials to be used as anonymous bids in auctions[2]. The AStERISK smart contract defines six functions (Setup, Create, Deposit, Commit, Reveal, Withdraw):

❖ **Setup:** A set of authorities jointly create an instance of the AStERISK contract by providing their public keys as well as any other scheme parameters as the number of authorities and the threshold parameter related to Coconut. This function can be run multiple times, and by different sets of authorities; the workers then select the set of authorities they trust upon executing Create.

❖ **Create:** Any 3rd party worker creates an auction by specifying the set of authorities trusted to issue credentials, as well as any application specific parameter or policy. They also specify, a commitment to the minimum price for which they are willing to operate; and two time-tamps[3], t_{commit} and t_{reveal} (where $t_{commit} < t_{reveal}$) used during the auction phase (see Section IV-B).

❖ **Deposit:** Bidders deposit v coins into a buffer account specified by the smart contract to request a credential on the public attribute v, and on a private randomly generated sequence number s. To prevent tracing traffic analysis, v should be limited to a specific set of possible values, similar to cash denominations. Each authority monitors the AStERISK smart contract, and issues a partial credential to the user—either on chain or off-chain—upon detecting a credential request (credential requests are processed only if bidders paid a deposit of v coins to the smart contract). Bidders locally aggregate all partial credentials into a consolidated credential.

❖ **Commit:** Bidders submit a bid by showing a valid credential to the smart contract; they also provide a proof of knowledge of the sequence number s along with a group element ζ uniquely built from s. If the proof and the credential check is passed, the smart contract records ζ. The group element ζ embeds the sequence number s and is therefore bound to the credential and the number of coins v it embeds—showing ζ effectively commits to v. The smart contract accepts the bidders input only before t_{commit}.

❖ **Reveal:** Bidders reveal v, ζ, and the credentials, as well as a proof that v is correctly embedded in the credentials and asserting correctness of ζ. The smart contract accepts the bidders input only after t_{commit} and before t_{reveal}, and if the smart contract previously recorded ζ (*i.e.,* if the bidder committed to its bid). Workers open the commitment to the minimum price they committed to during Create.

❖ **Withdraw:** To withdraw the coins, the bidder provides the smart contract with a zk-proof of knowledge of their private

key by binding the proof to the address $addr$ where they wish to redeem the coins; they also provide the consolidated credential, ζ, and a zk-proof attesting its correctness. To prevent double spending, the contract keeps a record of all group elements ζ that have already been shown. Upon showing a ζ embedding a fresh (unspent) sequence number s, the contract verifies the credential and zero-knowledge proofs, and that ζ does not already appear in the spent list. Then, it withdraws v coins from the buffer, sends them to the specified address $addr$, and adds ζ to the spent list. Bidders can only withdraw coins after t_{reveal}, and only credentials that have been recorded by Commit and Reveal can be used to withdraw coins (this effectively locks the funds of bidders that deviate from the protocol). According to the policy set upon executing Create, the winner of the auction may be treated differently.

❖ **SubmitWork:** After t_{reveal}, the winner contacts the worker through a private channel, and uploads cryptographic material to the smart contract allowing the worker to retrieve its payment. Winners can anonymously prove they actually won the auction by proving possession of the consolidated credential, ζ, and a zk-proof attesting its correctness. In the case of Filecoin, for example, the winner proves they won the auction to the smart contract, and uploads a hash of the encrypted file to store along with a signature. The file hash is then used to verify Proofs of Spacetime [15] submitted by the worker and release payments from the bidder's deposit over time.

B. The AStERISK Protocol

Figure 1 shows the execution of an auction on AStERISK; the auction is divided in three phases:

- **Preparation phase (Figure 1a):** Bidders pay deposits and retrieve a credential required to participate in auctions, and storage nodes submit their offers.
- **Auction phase (Figure 1b):** Bidders commit and later reveal their bids. The smart contract determines the winner by applying a Vickrey auction mechanism [28].
- **Execution phase (Figure 1c):** Auction winner contacts its corresponding storage nodes, prove their identity and submit files to store. Storage node receives rewards for storing files over time.

a) Preparation phase: A set of authorities executes the Setup function of the AStERISK smart contract described in Section IV-A. Any worker executes Create (Figure 1a-❸) to create an auction by specifying the auction parameters, and which set of authorities they trust. They also specify the two time-tamps, t_{commit} and t_{reveal} determining the auction's time line; workers advertise their product and provide a commitment to a minimum price v_0 at which they are willing to operate. Bidders execute Deposit by paying a deposit of v coins into a buffer account specified by the AStERISK smart contract (Figure 1a-❶), and retrieve a credential required to participate in auctions (Figure 1a-❷). The value v represents the number of coins they wish to bid.

b) Auction phase: Bidders execute Commit (Figure 1b-❶) to commit to a bid for a particular auction on the smart contract. Bidders' commitments are only considered valid if submitted before the deadline t_{commit}. Next, workers open the commitment to their minimum price, and bidders call

---

[2]The tumbler application is described at Section V.A of Coconut [22].

[3]Time may be defined as the number of blocks built on top of the main chain of the blockchain.
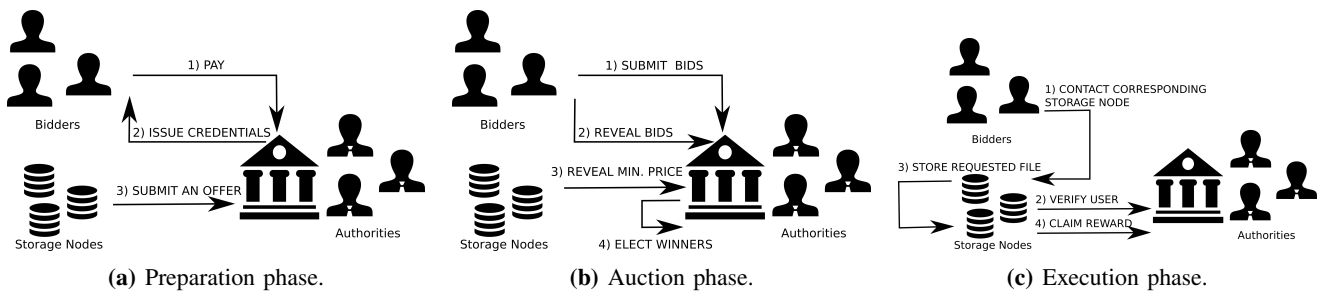
(a) Preparation phase. **(b)** Auction phase. **(c)** Execution phase.

Fig. 1: Overview of auction executions on AStERISK

Reveal (Figure 1b-❷) to open their commitment to the smart contract; bidders openings are valid only if submitted before the deadline t_{reveal}. Finally (Figure 1b-❸), the winner of the auction is deduced from the execution trace of the smart contract. The group element ζ associated with the highest v (if $v \geq v_0$) indicates the winner of the auction—all bidders that correctly followed the protocol (except the winner) may withdraw their coins calling the Withdraw function of the AStERISK contract. If there is no winner, the auction fails and every bidder may withdraw their money. AStERISK applies a Vickrey auction mechanism [28]; the winner of the auction is the bidder with the highest bid v, and pays the price of the second highest bid, v'. This, in turn, means that the winner is free to call Withdraw to withdraw $(v - v')$ coins.

c) Execution phase: The winning bidder executes SubmitWork (Figure 1c-❹) and submits hashes of the encrypted files to store. They then contact the corresponding worker off-chain, prove they are the rightful winner by showing their credential, and directly transmit the file replica $\mathcal{R}_{ek}^{\mathcal{D}}$ to the worker. The worker can then start generating proofs of useful work, submit them to the blockchain, and claim rewards from the winner's deposit. Verifying the proof of useful work and releasing the payment is an integral part of the underlying blockchain, and is out of the scope of this work.

C. Discussion on Auction Specifics

AStERISK involves worker specific contracts that offer resources in the form of a single item. In practice more sophisticated auctions are of interest where multiple workers offer their resources in the form of multiple items while bidders express their bids for a subset of them on the same contract; such ambition comes with the challenges of (*i*) auction execution, and (*ii*) bidding privacy and security.

With respect to the auction execution, it has been shown that combinatorial auctions can be modelled similarly to the set packing problem, meaning that they are NP-hard and there is no polynomial-time algorithm for finding the optimal allocation. A solution would be to consider a system where bidders are associated with up to a single item. That would lead to a computationally feasible and efficient assignment, *i.e.,* multi-item unit demand auction [29] setting, on the expense however of bidders' flexibility in expressing their preferences. On the other hand, the bidders are expected to submit a vector of bids, *i.e.,* one bid for each offered item. Although the Deposit function can be easily scaled up, *i.e.,* by putting as a deposit the sum of bids or the maximum bid in a vector, the analysis of such vectors of bids could reveal information

on the bidder and create a privacy challenge that has to be addressed.

V. PROTOCOL ANALYSIS

In this section we provide details of how AStERISK achieves the design goals described in Section III.

a) Hidden Minimum Price: Upon the execution phase of the auction, AStERISK guarantees that a worker will offer their resources at a price higher than the minimum price or not offer their resources at all.

b) Bidders privacy: AStERISK takes advantage of Coconut's unlinkability property to break the link between the deposit of coins, the commit of bids, and the withdraw of the coins. As a result, bidders can submit bids on auctions without revealing their identity, yet proving possession of a valid credential.

c) Bids privacy: Bids are kept private until t_{commit}; the zero-knowledge property of Coconut credentials implies that no information about v is revealed while committing to a bid (Figure 1b-❶).

d) Bids binding: Bidders are bound to their bids as they are first required to commit to their bid (Figure 1b-❶), and then to open the commitment (Figure 1b-❷). Bidders cannot open the commitment to another value than the previously committed as this implies forging the Coconut credentials.

e) Public auditability: AStERISK is implemented as a smart contract; its correct execution can be verified by any 3rd party by taking advantage of the public auditability of the underlying smart contract platform.

f) Fairness: No single authority can create credentials and steal all the coins in the buffer account of the smart contract—the threshold property of Coconut implies that adversaries need to corrupt an arbitrarily large set of authorities for this attack to be possible. Bidders cannot participate to the auction phase (Figure 1b) without paying a deposit to receive a valid credential; setting $t_{commit} < t_{reveal}$ forces the bidders to commit to their bid before revealing it, preventing any 3rd party from seeing other bidder's bid before committing to a value. Bidders dropping out after committing a bid (and never revealing it) are financially penalized as they cannot withdraw their coins. Coconut provides blind issuance which allows bidders to obtain a credential on the sequence number s without the authorities learning its value. Without blindness, any authority seeing s could potentially race the bidders, and withdraw the coins of the credential—blindness prevents authorities from stealing the coins. Keeping a spent list of all group elements ζ prevents double-spending attacks [27] without revealing the sequence number s; this prevents an

AStERISK Chainspace smart contract			
Operation	μ **[ms]**	$\sqrt{\sigma^2}$ **[ms]**	**size [kB]**
Create [g]	28.433	± 0.214	∼ 1.8
Create [c]	0.0148	± 0.002	-
Commit [g]	194.243	± 0.410	∼ 2.7
Commit [c]	355.852	± 15.880	-
Reveal [g]	205.656	± 5.659	∼ 2.7
Reveal [c]	351.192	± 8.514	-
Withdraw [g]	188.925	± 2.084	∼ 2.6
Withdraw [c]	336.533	± 4.490	-
SubmitWork [g]	197.399	± 6.537	∼ 2.7
SubmitWork [c]	368.948	± 13.116	-

TABLE I: Timing and transaction size of the AStERISK Chainspace smart contract (described in Section IV-A), measured over 10,000 runs. The transactions are independent of the number of authorities. The notation [g] denotes the execution of the procedure and [c] denotes the execution of the checker. The Deposit function is not implemented as it is identical to the tumbler application described in Coconut [22].

attacker from exploiting a race condition upon withdrawing and which may lock bidders coins.

g) Non-interactivity: Bidders do not interact with each other. During the preparation phase (Figure 1a), Bidders only interact with a subset of the the authorities to receive a credential; during the auction phase (Figure 1b), they only interact with the smart contract; during the execution phase (Figure 1c), bidders only interact with the workers or with the smart contract to withdraw their coins.

h) Censorship resistance: The decentralized nature of the underlying smart contract platform makes the AStERISK smart contract resilient to censorship. Furthermore, a small subset of authorities cannot block the issuance credentials—the service is guaranteed to be available as long as at least a threshold number of authorities are running.

i) Distributed authority: AStERISK introduces no single trusted 3rd party; the AStERISK contract is executed on a decentralized smart contract platform, and Coconut allows threshold issuance of credentials.

j) Auction's economic properties: An auction satisfies all those properties only under the condition of *price-taker participants* [30], *i.e.,* both bidders and workers have no impact on the auction prices. In the single item auctions we consider here, it has been proven that Vickrey auctions possesses all these desired attributes based on the assumption of "sealed bids", where neither bidders nor workers have information about the state of the auction [28]. AStERISK through its privacy properties provides a technical implementation of the "sealed bids" assumption which prevents price manipulations.

VI. IMPLEMENTATION & EVALUATION

We provide an open-source implementation[4] of the AStERISK smart contract presented in Section IV-A for Chainspace [25]. Our implementation does not enforce conditions on timers t_{commit} and t_{reveal} as Chainspace currently does not provide functions to check block heights. We wrote our prototype in about 450 lines of Python code using an open source Python implementation of Coconut[5], which relies on petlib[6] and pblib[7]; the bilinear pairing is defined over the

---

[4]https://github.com/asonnino/coconut-chainspace

[5]https://github.com/asonnino/coconut

[6]https://github.com/gdanezis/petlib

[7]https://github.com/gdanezis/bplib

Barreto-Naehrig curve[8], using OpenSSL as arithmetic backend. Table I provides the timing and transaction size for each function of the smart contract; each experiment is the result of 100 runs measured on a commodity laptop (a MacBook Pro 13' 2.7 GHz Intel Core i7, running macOS Mojave). As expected, both the procedure and the checker of Create are extremely fast as they do not involve cryptographic operations. The checker of Commit, Reveal, Withdraw, and SubmitWork take on average the same (and the longest) time as their core operation is to verify credential validity; verifying credentials takes the longest time due to pairing operations [22]. The Deposit function is not implemented as it is identical to the tumbler application described in Coconut [22].

VII. RELATED WORK

There are several frameworks that target hiding private data submitted to a public ledger. Hawk [33] divides Smart Contract into public and private parts and secure private input using zero-knowledge proofs, but requires a centralized trusted manager to operate. Furthermore, ShadowEth [31] allows processing confidential Smart Contract data using Trusted Execution Environments (TEE). However, such a scheme requires users to trust the hardware vendor and can expose the system to TEE's vulnerabilities [37], [38]. On the subject of sealed-bid auctions, Blass and Kerschbaum [34] proposed Strain, that preserve bids privacy against malicious participants. Strain uses a two-party comparison protocol, but has a flaw that reveals the order of bids. Furthermore, running protocols involving MPC on blockchain is not efficient due to extensive computations and the number of rounds involved. Furthermore, Galal and Youssef [39] presented a protocol that ensures public verifiability, privacy of bids, and fairness. However, the solution scales badly with the number of bidders and relies on a random number retrieved from blockchains that are not proven to be secure. This scheme was improved in [35] using zk-SNARKS, but still relies on a centralized party for zero-knowledge proofs and does not protect bidders identity. An alternative approach was proposed by Bogetoft *et al.* [36]. The system uses a multiparty computation to perform auctions on encrypted bids. However, such a scheme reveals final assignment between bidders and objects and lacks transparency. Currently, Filecoin [16] does not implement an automated system assigning clients to storage nodes. Users are required to chose storage nodes manually and offers are publicly posted on the blockchain. Other industrial systems such as Golem [6], iExec [8] or SONM [7] either do not specify their requester-worker assignment technique or rely on similar, insecure solutions. AStERISK can be a useful component for all those platforms to increase their level of security and automatically determine optimal price for services. We summarize the previously discussed solutions and their security features in Table II.

VIII. LIMITATIONS

AStERISK inherits several limitations of Coconut which acts as the underlying credential scheme. These limitations are beyond the scope of this work, and deferred to future work. AStERISK is vulnerable if more than the threshold number of authorities are malicious; colluding authorities could create credentials to steal all the coins in the buffer of the smart

---

[8]https://tools.ietf.org/id/draft-kasamatsu-bncurves-01.html

System	Bids Privacy	Bidders Privacy	Bidders Non-Interactivity	Distributed Authority	Trusted Hardware	Public Auditability
ShadowEth [31]	✓	✗	✓	●	Intel SGX [32]	✓
Hawk [33]	✓	✗	✓	◖	None	✓
Strain [34]	✓	✗	✗	○	None	✓
Galal *et al.* [35]	✓	✗	✗	○	None	✓
Bogetoft *et al.* [36]	✓	✗	✓	●	None	✗
Filecoin [16]	✗	✗	✗	●	None	✓
AStERISK	✓	✓	✓	●	None	✓

TABLE II: Comparison of security properties achieved by different systems discussed in this section. The decentralization property reads as follows; ○ : relies on a trusted 3rd party, ◖ : relies on a trusted 3rd party for only one (or some) of the properties described in Section III, ● : does not rely on any trusted 3rd party.

contract. Note that bidders ' privacy is still guaranteed under colluding authorities, or an eventual compromise of their keys.

The smart contract implementation of AStERISK described in Section IV-A does not scale to a large number of users as each commitment and bid is kept on-chain. A potential solution is to defer the auction logic to a state channel [40], and only seal the final result of the auction on the blockchain. Moreover, AStERISK inherits from any scalability limitation of the underlying smart contract platform.

The winner of the auction may invoke SubmitWork, but refuse to transfer data to the worker. This prevents the worker from claiming the reward and leave their resources unused until the next auction. This issue is inherited from Filecoin and can be mitigated with additional mechanisms assuring fair exchange of digital goods [41].

IX. CONCLUSION AND FUTURE WORK

We presented AStERISK— a system for determining optimal prices for services in a shared economy environment and automatic assignments of requesters to the most optimal workers. AStERISK allows to securely perform sealed-bid auction on a blockchain using anonymous credentials and zero-knowledge proofs of knowledge. Our system allows workers to specify a minimal price for their services, protects users' bids as well as the identity of the bidders. Contrary to the previous work, AStERISK does not rely on a trusted 3rd party to issue credentials, but rather on a set of entities that can be freely chosen by users. The distributed authorities issue only partial credentials that are merged locally by each user protecting the system from a subset of malicious authorities. We showed how AStERISK can be deployed in the Filecoin network and adapted to other shared economy systems operating on blockchain. As a part of future work, we plan to extend our system to securely support requesters' evaluation of multiple items and protect against data analysis attacks. Furthermore, we will investigate the scalability of our solution with large networks and better integration with additional platforms.

ACKNOWLEDGEMENTS

REFERENCES

[1] J. Swearingen, "http://nymag.com/selectall/2018/03/when-amazon-web-services-goes-down-so-does-a-lot-of-the-web.html," 2018.

[2] A. Glaser, "How apple and amazon are aiding chinese censors," 2018.

[3] B. Debatin, J. P. Lovejoy, A.-K. Horn, and B. N. Hughes, "Facebook and online privacy: Attitudes, behaviors, and unintended consequences," *Journal of computer-mediated communication*, vol. 15, no. 1, pp. 83–108, 2009.

[4] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[5] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[6] T. G. Project, "Golem whitepaper." https://golem.network/doc/Golemwhitepaper.pdf, 2016.

[7] "Sonm documentation." https://docs.sonm.com/, 2018.

[8] "iexec whitepaper." https://iex.ec/whitepaper/iExec-WPv3.0-English.pdf, 2018.

[9] A. Angelo, P. Thellmann, and D. Dalkilic, "Rewarding the token economy.." https://bounty0x.io/whitepaper_en.pdf, 2018.

[10] Y. Combinator, "Ipfs, coinlist, and the filecoin ico with juan benet.."

[11] P. Labs, "Proof of replication technical report," tech. rep., 2017.

[12] M. Al-Bassam, A. Sonnino, M. Król, and I. Psaras, "Airtnt: Fair exchange payment for outsourced secure enclave computations," *arXiv preprint arXiv:1805.06411*, 2018.

[13] M. Król and I. Psaras, "Spoc: Secure payments for outsourced computations," *Proc. NDSS Workshop on Decentralized IoT Security and Standards, San Diego, CA*, 2018.

[14] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz, "Permacoin: Repurposing bitcoin work for data preservation," in *2014 IEEE Symposium on Security and Privacy (SP)*, pp. 475–490, IEEE, 2014.

[15] J. Benet, D. Dalrymple, and N. Greco, "Proof of replication," tech. rep., Technical report, Protocol Labs, July 27, 2017. https://filecoin. io/proof-of-replication. pdf. Accessed June, 2018.

[16] P. Labs, "Filecoin: A decentralized storage network," tech. rep., 2018.

[17] V. Buterin *et al.*, "Ethereum white paper," 2013.

[18] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, "Quadratic span programs and succinct nizks without pcps," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 626–645, Springer, 2013.

[19] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, "Snarks for c: Verifying program executions succinctly and in zero knowledge," in *Advances in Cryptology–CRYPTO 2013*, pp. 90–108, Springer, 2013.

[20] M. Chase, S. Meiklejohn, and G. Zaverucha, "Algebraic macs and keyed-verification anonymous credentials," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1205–1216, ACM, 2014.

[21] D. Pointcheval and O. Sanders, "Short randomizable signatures," in *Cryptographers' Track at the RSA Conference*, pp. 111–126, Springer, 2016.

[22] A. Sonnino, M. Al-Bassam, S. Bano, and G. Danezis, "Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers," *arXiv preprint arXiv:1802.07344*, 2018.

[23] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 514–532, Springer, 2001.

[24] M. Castro, B. Liskov, *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, pp. 173–186, 1999.

[25] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis, "Chainspace: A sharded smart contracts platform," *arXiv preprint arXiv:1708.03778*, 2017.

[26] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 583–598, IEEE, 2018.

[27] G. O. Karame, E. Androulaki, and S. Capkun, "Double-spending fast payments in bitcoin," in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 906–917, ACM, 2012.

[28] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," *The Journal of finance*, vol. 16, no. 1, pp. 8–37, 1961.

[29] T. Andersson and A. Erlanson, "Multi-item vickrey–english–dutch auctions," *Games and Economic Behavior*, vol. 81, pp. 116–129, 2013.

[30] R. B. Myerson and M. A. Satterthwaite, "Efficient mechanisms for bilateral trading," *Journal of economic theory*, vol. 29, no. 2, pp. 265–281, 1983.

[31] R. Yuan, Y.-B. Xia, H.-B. Chen, B.-Y. Zang, and J. Xie, "Shadoweth: Private smart contract on public blockchain," *Journal of Computer Science and Technology*, vol. 33, no. 3, pp. 542–556, 2018.

[32] V. Costan and S. Devadas, "Intel sgx explained.," *IACR Cryptology ePrint Archive*, vol. 2016, no. 086, pp. 1–118, 2016.

[33] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *2016 IEEE symposium on security and privacy (SP)*, pp. 839–858, IEEE, 2016.

[34] E.-O. Blass and F. Kerschbaum, "Strain: A secure auction for blockchains," in *European Symposium on Research in Computer Security*, pp. 87–110, Springer, 2018.

[35] H. S. Galal and A. M. Youssef, "Succinctly verifiable sealed-bid auction smart contract," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pp. 3–19, Springer, 2018.

[36] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, *et al.*, "Secure multiparty computation goes live," in *International Conference on Financial Cryptography and Data Security*, pp. 325–343, Springer, 2009.

[37] S. Davenport, "Sgx: the good, the bad and the downright ugly." https://www.virusbulletin.com/virusbulletin/2014/01/sgx-good-bad-and-downright-ugly, 2014.

[38] L. M. JP Aumasson, "Sgx secure enclaves in practice security and crypto review." https://www.blackhat.com/docs/us-16/materials/us-16-Aumasson-SGX-Secure-Enclaves-In-Practice-Security-And-Crypto-Review.pdf, 2016.

[39] H. Galal and A. Youssef, "Verifiable sealed-bid auction on the ethereum blockchain," in *International Conference on Financial Cryptography and Data Security, Trusted Smart Contracts Workshop. Springer*, 2018.

[40] A. Miller, I. Bentov, R. Kumaresan, and P. McCorry, "Sprites: Payment channels that go faster than lightning," *CoRR abs/1702.05812*, 2017.

[41] S. Dziembowski, L. Eckey, and S. Faust, "Fairswap: How to fairly exchange digital goods," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 967–984, ACM, 2018.