Block Auction: A General Blockchain Protocol for Privacy-Preserving and Verifiable Periodic Double Auctions

Theodoros Constantinides* and John Cartlidge†
Department of Computer Science
University of Bristol, Bristol, UK
*Email: *tc17231@bristol.ac.uk, †john.cartlidge@bristol.ac.uk*

***Abstract*—We use blockchain technology to tackle the problem of securing periodic double auctions for financial 'dark pool' trading, such that the privacy of pre-trade order information is preserved and the behaviour of the auction operator can be verified. A smart contract is used as a public audit trail without revealing the order intention of traders. Auction matching is performed off-chain, allowing alternative auction mechanisms to be used without altering the smart contract code. A full system implementation is deployed on the Harmony blockchain, and comparative evaluation of the protocol demonstrates some clear advantages over the closest published alternative.**

***Index Terms*—blockchain, double auction, privacy, smart contract**

1. Introduction

Dark pools are financial trading venues where all order information is non-displayed (i.e., hidden from view). Dark pools are principally designed for large volume traders to avoid market impact, a costly effect where the market moves adversely when large volume orders are observed on a displayed (or 'lit') order book used by an exchange. However, the lack of transparency in dark pools is a concern to regulators and, in 2018, European regulators introduced rules to limit the amount of dark pool trading [1]. This has resulted in a rise in popularity of 'semi-transparent' periodic auctions, where non-displayed orders are matched at regular intervals. In dark pools and periodic auctions, market impact is only avoided if order information is not leaked or misused, either accidentally or maliciously. Unfortunately, on multiple occasions, dark pool operators have misused order information or altered the auction rules, resulting in operators paying hundreds of millions of dollars in fines [2].

Contribution: We present a novel smart contract protocol for implementing verifiable and privacy-preserving periodic double auctions on a blockchain. Under the protocol, traders first place encrypted order commitments that cannot be read by the operator. At the start of auction matching, orders are revealed to the operator and the results of the auction (which is run off-chain, thereby allowing *any* auction mechanism to be used without altering the smart contract) are published. This provides traders with provable guarantees about their order execution and eliminates front-running and mechanism misuse by the operator. We implement the smart contract and demonstrate its efficiency and flexibility.

. Accepted manuscript, to appear in the *International Symposium on Fintech and Blockchain Systems* at the *4th IEEE International Conference on Blockchain* (IEEE Blockchain-2021), Dec. 2021.

2. Related Work

In standard auctions, buyers compete to purchase an item by submitting a 'bid'. The highest bidder wins the auction and pays a price determined by the auction rules. The most common auction types include sealed-bid (where all bids are private) and open format (where all bids are observable). Double auctions are more complicated, and bring buyers and sellers together to transact some item. Orders from buyers ('bids') and sellers ('asks') are submitted and a clearing price (or transaction price) is determined by the double auction rules. Double auctions can be periodic (orders are submitted during an entry phase and then clearing takes place to match those orders after some fixed interval), or continuous (orders can be submitted and matched at any time). Previous work on privacy-preserving sealed-bid auctions have been surveyed at length elsewhere [3]. Here, we focus our review on privacy-preserving double auctions.

In 2006, privacy-preserving double auctions were first practically considered by Bogetoft et al. [4]. Their protocol executed a one-shot periodic double auction between three MPC parties, enabling buyers and sellers to trade multiple units securely. Subsequently, in 2008, the double auction protocols of Bogetoft et al. were deployed to secure the annual Danish sugar beet auction; becoming the first real-world implementation of a secure MPC auction [5]. In 2015, Jutla [6] introduced protocols for repeated periodic double actions using MPC, and suggested that a regulatory authority could be included as an MPC party to reduce the likelihood of collusion between operators. In 2019, Cartlidge et al. [7] presented 2-party and 3-party MPC protocols for securing dark pool trading venues. Three common mechanisms – continuous double auction, periodic auctions, and a periodic volume match – were empirically evaluated, with results suggesting that the simplest volume match algorithm has performance that could handle throughput required for a real-world dark pool. In 2020 [2], the same authors extended their work by implementing a secure MPC approximation of the trading mechanism used by Turquoise Plato, Europe's largest dark pool. Their protocols were evaluated on simultaneous trading across 4000 instruments, demonstrating that MPC approaches are finally capable of practical application in securing financial trading venues.

In 2007, Thorpe and Parkes [8] presented a protocol for a continuous double auction mechanism that uses homomorphic encryption along with a bulletin board to record all activity. The purpose of the protocol is to ensure provably correct behaviour of the auction operator. Traders encrypt orders using the operator's public key and post them to the bulletin board, where they are timestamped and appended with a unique identifier. The operator then decrypts and matches the orders, before updating the bulletin board with details of the trades, along with proofs of correctness so that anyone can verify the operator's actions. In 2009, the same authors extended their work by proposing a protocol that uses a commitment scheme to implement a combinatorial securities exchange [9]. As suggested in [8], these protocols can further improve security guarantees by using trusted computing infrastructure. However, the issue of trusting the trusted infrastructure remains.

A number of systems remove the need for a trusted third party by implementing double auctions using blockchain technology. Published between 2017-19, the main use case of these works are electricity trading and smart grids [10], [11], [12], [13], [14]. However, none of these systems are concerned with keeping order information secret and therefore each acts as a lit market rather than a dark market.

Several works consider both MPC and blockchain approaches for securing double auctions. In 2017, Zhang and Wang [15] introduced the Republic Network (REN); a decentralised dark pool protocol that uses secure MPC to provide a platform for trading cryptocurrencies and digital assets. Also in 2017, Massacci et al. [16] introduced a secure futures exchange (FuturesMEX) using a distributed ledger, MPC, and non-interactive zero-knowledge proofs on committed inputs. The FuturesMEX protocol aims to replicate the main functionalities of a futures exchange, such as the Chicago Mercantile Exchange, and is implemented as a continuous double auction that is run by a centralised clearing house. In 2020, Liu et al. [17] proposed a blockchain-based fair and secure electronic double auction protocol (BFSDA) that used MPC to guarantee fairness and improve security.

In 2021, Ngo et al. [18] introduced a protocol for decentralised over-the-counter (OTC) dark pools using a distributed ledger and zk-SNARKs. Using Witness Key Agreement (WKA), two parties are able to agree on a shared key only if committed information meets a desired condition. WKA enables traders to negotiate via a public bulletin board (a blockchain), sending messages securely until a trade is agreed. A theoretical estimate of the performance of the system suggested each WKA operation takes less than 15s, which would enable a throughput of messages expected to result in approximately 50 trades/day, as seen in some real world OTC markets. Also in 2021, Galal and Youssef [19] introduced a commitment scheme with zero-knowledge proofs to achieve verifiable and secrecy-preserving double auctions.

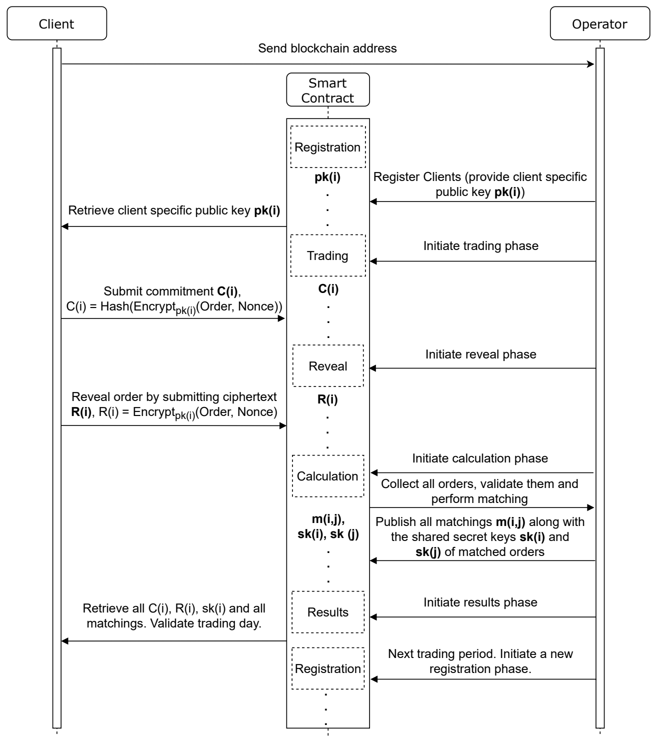
Client
Operator
Send blockchain address
Smart Contract
Registration
pk(i)
Register Clients (provide client specific public key pk(i))
Retrieve client specific public key pk(i)
Trading
Initiate trading phase
Submit commitment C(i), C(i) = Hash(Encrypt_pk(i)(Order, Nonce))
C(i)
Reveal
Initiate reveal phase
Reveal order by submitting ciphertext R(i), R(i) = Encrypt_pk(i)(Order, Nonce)
R(i)
Initiate calculation phase
Calculation
Collect all orders, validate them and perform matching
m(i,j), sk(i), sk (j)
Publish all matchings m(i,j) along with the shared secret keys sk(i) and sk(j) of matched orders
Results
Initiate results phase
Retrieve all C(i), R(i), sk(i) and all matchings. Validate trading day.
Registration
Next trading period. Initiate a new registration phase.

Figure 1. System functionality overview, showing how the operator and a client interact with the smart contact during each protocol phase.

3. Protocol Overview

We introduce a novel protocol for periodic double auctions that allows traders to trade *secretly* (no order information is revealed prior to the auction) and *verifiably* (users can verify that the auction protocol has executed correctly).[1] The key actors of the system are (i) the operator, who runs the auction, (ii) the clients, who use the system to trade assets, and (iii) the smart contract, which acts as a public bulletin board where clients submit orders and the operator publishes results. An overview is presented in Figure 1. In summary, for each auction period, (i) client traders register, (ii) clients send secret commitments (orders) to the smart contract, (iii) at the start of the matching period, all orders are revealed to the operator and the matching mechanism is performed off-chain, (iv) the operator publishes auction results (matched orders) to the smart contract, and (v) finally results can be verified by the clients to ensure that the auction mechanism was performed correctly. Note that since the auction matching is performed off-chain, any matching mechanism can be used by the operator, making the framework generally applicable for any periodic double auction without changing

1. A full system implementation is available open source at the project's github repository: https://github.com/theodoros701/BlockAuction.

the smart contract. Also, while the encryption scheme used in the prototype is ECIES using secp256k1 and AES-256-GCM, any other similar public key encryption scheme can be substituted without changing the smart contract.

The system works like a commitment scheme with some additional properties. The operator first registers clients on the smart contract by assigning a specific public key $pk(i)$ to the address of each client, i. Client i then generates orders of the form $O(d, a, p, v, m)$, where d is the direction (to buy or sell), a is the asset, p is limit price, v is volume, and m the minimum execution size. A 32-byte random bit-string $nonce$ is then appended to order O and the result is encrypted using the operator's public key, giving $\{O, nonce\}_{pk(i)}$. Each client then publishes commitment $C(i) = Hash(\{O, nonce\}_{pk(i)})$ on the smart contract, which ensures that the order is not revealed, not even to the operator. At the end of the trading phase, clients reveal orders only to the operator by publishing the ciphertext of their order, $R(i) = \{O, nonce\}_{pk(i)}$. At this point, anyone can verify that the published ciphertexts match the commitments. To incentivise clients to reveal orders, a deposit is provided by the client when commitment $C(i)$ is sent; and deposits are returned when reveal $R(i)$ is sent. The system can handle trading in any number of assets during the same trading period. This has the advantage of ensuring that the asset a client wants to trade is not revealed during order submission. However, to simplify the implementation, we only allow clients to submit one order per auction period. Therefore, multiple assets can only be traded by a client over multiple auction periods. This simplification can easily be relaxed in future to allow either multiple orders each period or an order type that combines multiple assets.

At the end of the reveal phase, the operator decrypts each valid order and then matches the plaintext orders off-chain, using a clearing mechanism that is publicly known to the clients. We designed the system in this way so that the smart contract is agnostic of the order clearing algorithm. This makes the protocol modular and customisable, as new matching algorithms can be added on demand without needing to alter and redeploy the smart contract. In the software implementation we demonstrate three auction clearing algorithms: (i) single clearing price that maximises volume traded; (ii) 'maximised matching' [20] where orders are cleared at different prices to maximise volume traded; and (iii) a simple 'volume match' [7] where orders have price value $p = null$ and all trades execute at a price taken from some external reference (e.g., the primary exchange).

Once matching is completed, the operator publishes the results on the smart contract, revealing only the orders that were matched (i.e., those that resulted in a trade), along with the shared secret key $sk(i)$ that was used to encrypt the order. At this point, anybody can verify that the published plaintext orders match the ciphertext orders published earlier. Unmatched orders (i.e., those that did not trade) remain secret to anyone apart from the client who submitted the order and the operator. Any observer, including the clients, can verify the published results to make sure that the operator conducted the auction honestly. For client j with unmatched order O_j, j can verify that O_j was not competitive enough to execute by combining their private information about O_j with the public information in the smart contract. If a client detects that something went wrong, they can notify the other clients through the smart contract. Since every client is incentivised to verify the auction for themselves and notify others if the operator acted dishonestly, the clients as a collective can verify the execution of the whole auction. This implies that if a client verifies the auction with the information they have available and also do not hear that another client had a failed verification, then they can be confident that the operator was honest. Where dishonesty is shown, the public record in the blockchain can be used as evidence to prove a legal claim against the operator (e.g., for precedent see [21]).

Currently, the protocol provides no way for an individual to check the execution by themselves, without relying on the collaborative checking of all participants. This can be considered a limitation of the system, as some clients might fail to perform their verification correctly or might choose to omit verification entirely. However, this behaviour is most likely to affect parties that failed to verify the auction as other parties will detect fraud against them (as is the case for the single clearing price periodic auction for example). Yet, given that the system is flexible enough to allow the operator to run any auction matching algorithm, there might be cases where parties that verified the auction are affected if others fail to do so. One way to prevent this is for the operator to publish proofs of the correct execution of the protocol, along with the results. We have not taken this approach, but possible ways to implemented such proofs include MPC in the head or zkSNARKs. In theory, both methods should allow an operator to prove to anyone, in zero knowledge, that the execution of the protocol was correct.

3.1. Harmony Blockchain Deployment

Ethereum is the primary smart contract blockchain and offers the security of a large network and the maturity and availability of development tools, resources, and community. However, Ethereum's Proof-of-Work consensus mechanism makes it relatively slow (13 seconds per block) and expensive.[2] Therefore, Harmony [22] blockchain was selected for system deployment. Harmony is EVM-compatible, has an Ethereum-like API, and uses Proof-of-Stake for fast consensus and negligible transaction costs.[3] Therefore, migrating the smart contract between Etherium and Harmony is straightforward. In future, for enterprise use, we suggest that the protocol can be deployed on Hyperledger Fabric [23], as the protocol performs like a permissioned system.

2. Gas cost estimates using ether prices in July 2021 suggest one auction with 100 clients and 25 executed orders ($\sim 29M$ gas) would cost (combined for clients and operator) in the region of \$1740 (0.87 ether).

3. Running the same auction in Harmony costs \$0.02 (0.29 ONE).

TABLE 1. EXPERIMENTAL GAS COSTS FOR OPERATOR AND CLIENTS.

Operator	Cost	Client	Cost
Deploy contract	3.70M	Submit order	52k (67k)
Register client	87k	Reveal order	171k
Initiate trading phase	[29k, 94k]	Change order	30k
Change phase	[31k, 38k]	Delete order	22k (19k)
Publish match	95k or 149k	Claim fraud	25k
Assign new public key	38k		
Delete order	[20k, 37k]		
Delete client	[21k, 48k]		

Ranges given where operations depend on other parameters.
Parentheses indicate special case where memory is allocated/released.

TABLE 2. THEORETICAL MINIMUM PHASE DURATION.

Phase	Minimum Duration (Blocks)
First Registration	$\lceil c/172 \rceil$
Trading	$\lceil c/288 \rceil$
Reveal	$\lceil c/88 \rceil$
Calculation (results publication)	$[\lceil e/201 \rceil,\ \lceil e/158 \rceil]$
Results (buffer phase)	1
Registration (reset - case 1)	$\lceil (c-e)/429\ +\ e/405 \rceil$
Registration (reset - case 2)	$\lceil (c-e)/429\ +\ e/200 \rceil$
Registration (reset - case 3)	$\lceil (c-e)/429\ +\ e/111 \rceil$
Registration (reset - case 4)	$\lceil (c-e)/115\ +\ e/111 \rceil$

c is number of clients, e is number of executed orders

4. Protocol Analysis

We analyse the protocol theoretically and empirically. We present theoretical results for an Ethereum implementation to enable comparison with other protocols presented in the literature. Gas costs are identical for Harmony as a similar pricing strategy is used. However, the results for duration require conversion as Harmony has a different block gas limit. This conversion is trivial and explained below.

4.1. Theoretical Analysis of Gas Usage

In Ethereum, every transaction or operation carries a gas fee. The gas cost of low-level operations is defined in the Ethereum yellow paper [24, Appendix G]. The transaction cost in *ether* depends on the gas cost of the transaction and the gas price specified by the sender. So, when analysing contracts, it is most sensible to consider gas cost (which are fixed) rather than the cost in ether (which vary). We empirically measured the gas cost of all contract functions.

There is a constant gas cost per client and a constant cost per matching for the operator and so the gas cost grows linearly on the number of clients and the number of matchings. The minimum cost of an iteration of the protocol for the operator equals the fixed costs to run the auction, plus the cost to publish order matchings and to delete clients' orders at the end. This can be estimated at:

$$\text{Gas} \approx F + 149000 \times m_2 + 95000 \times m_1 + 35000 \times c \quad (1)$$

where F accounts for fixed costs to switch between auction phases ($F \approx 200k$), m_2 is the number of matchings with two secret publications, m_1 is the number of matchings with one secret publication and c is the number of clients that submitted orders. The cost of publishing a matching depends on the number of orders that need their secret publication. If an order was partially matched in a previous matching, its secret does not need to be published again and so some gas is saved. In practice, most publications are m_1 because if two orders are matched and one has leftover volume, it is likely that the partially matched order will be matched again. However, this depends on the matching algorithm, so it may not be true in general. Contract deployment and client registration are only performed once, so they are not recurring costs and are not counted in the cost of an iteration. Gas costs for all operator functions are presented in the left hand side of Table 1.

The clients bear a considerably smaller gas cost. If a client does not participate in the auction, they will have no fees to pay. A client that participates in the auction must pay gas fees for submitting an order and then for revealing it. The minimum cost of participation for a client can be estimated at around $223k$ gas. Gas costs for all client functions are presented in right hand side of Table 1.

4.2. Theoretical Analysis of Runtime Duration

Due to the nature of blockchain systems, it is difficult to calculate the exact runtime of a protocol like the one presented here. Predicting exactly when a block gets mined is not possible and there are unpredictable network delays. For this reason, instead of giving a concrete runtime we provide the minimum runtime (in number of blocks) it takes to complete a task as a function of a variable quantity, like the number of clients or the number of executed orders (orders matched by the protocol). A runtime (in seconds) can then be estimated by multiplying the number of blocks by the estimated time it takes for a block to be mined, which for Ethereum is around 13 seconds and for Harmony is around 2 seconds. This is the best-case scenario, and in practice additional delays can be expected. Finality [25] should also be considered, as forks can take place and malicious parties might try to take advantage if the protocol does not wait for a certain number of confirmations. For Ethereum, it is recommended to add another 25 blocks on each of the phases, while Harmony finality takes only one block.

The smart contract acts as a state machine that keeps track of the auction phase. There are five distinct phases and so at least five blocks are needed for transitions. Assuming that each phase requires one block, the protocol at a minimum requires ten blocks to run. The Ethereum block gas limit is currently 15 million, and so to calculate the duration of a phase we divide 15 million by the gas cost of the operations performed in that phase. For example, submitting an order costs $52k$ gas so we calculate that a block can process $15,000,000/52,000 \approx 288$ submissions. Therefore, if c clients submit an order, we will need at least $\lceil c/288 \rceil$ blocks. Harmony has a larger gas limit of 80 million, so the results require adjustment. In practice, it is

TABLE 3. EXPERIMENTAL RESULTS USING HARMONY. TOP: SUM OF ALL PHASES; BOTTOM: INDIVIDUAL PHASES.

		Full Trading Period				Repeated Trading Period			
Clients	Matches	Cost	Duration (s)	Blocks	Transactions	Cost	Duration (s)	Blocks	Transactions
10	3	7,887,035	50	25	48	3,227,971	40	20	38
100	42	44,917,739	168	84	447	32,458,789	140	70	347
1000	704	438,954,083	1,818	909	4,709	348,362,209	1,556	778	3,709

		Registration		Trading		Reveal		Calculation			Order Deletion	
Clients	Matches	Cost	Blocks	Cost	Blocks	Cost	Blocks	Cost	Blocks	Process (s)	Cost	Blocks
10	3	0.96M	5	0.57M	5	1.74M	5	0.43M	4	0.79	0.46M	3
100	42	8.76M	14	5.24M	23	17.1M	20	5.45M	12	9.18	4.64M	12
1000	704	86.9M	131	51.9M	240	171M	209	78.3M	170	110.4	47.4M	155

highly unlikely that a single entity will be able to consume all the gas of a block, unless they mine it themselves, so duration will be higher as other people's transactions will be included in the same blocks. Finally, network delays and other factors are not considered, although in practice they can be considerable.

Table 2 presents theoretical minimum duration for each phase, assuming all clients submit and reveal orders. The first registration is where clients are registered, while the other four registration cases are alternative options for contract reset before the next trading period. Each case offers a different level of privacy, secrecy, and cost. Case 1 is the minimum where the operator simply deletes all orders. In case 2, the operator deletes all orders and assigns new public keys to clients whose orders were executed. This can be done if forward secrecy is a concern. In case 3, all orders and clients who took part are deleted, and those clients are registered again with a new address. This helps to preserve anonymity as an observer could track addresses to guess the identity of a client from published orders. Finally, case 4 provides perfect anonymity but is the most expensive as all clients are deleted and registered again.

In terms of number of transactions, the runtime of the protocol is easier to calculate: each client (i) is registered; (ii) submits an order; (iii) reveals the order; and finally (iv) the order is removed from the contract. As each operation takes one transaction, four transactions are needed per client. There are additionally five transactions to transition between the different phases of the protocol. Finally, there are the order matching publications which can be thought of as a variable quantity. This gives us an estimate of the runtime in number of transactions as:

$$4 \times c + 5 + m \tag{2}$$

where c is the number of clients and m the number of matching publications. If the operator wishes to assign new keys or delete and re-register the clients this number increases to:

$$5 \times c + 5 + m \tag{3}$$

as one extra transaction per client is needed. For repeated auctions, first client registration is not required again, giving:

$$3 \times c + 5 + m \qquad \text{or} \qquad 4 \times c + 5 + m \tag{4}$$

4.3. Empirical Analysis of Harmony Deployment

To empirically test the performance of the protocol, we ran three experiments on Harmony's testnet, with 10, 100, and 1000 clients.[4]

For each experiment: (i) the contract was deployed; (ii) all clients registered; and then (iii) one protocol iteration took place, during which each client submitted and revealed a random order; finally, at the end of the trading period, (iv) the operator published the results of the auction and the orders were deleted. A simple volume match algorithm [7] was used as the periodic auction mechanism.

Results are presented in Table 3. Top: we see that the number of transactions grows linearly with the number of clients, c, with results for the full trading period verifying equation (2) and results for a repeated trading period verifying equation (4). Empirical gas costs verify equation (1). Gas costs also increase linearly with c, apart from the full trading day for 10 clients where cost is skewed by the high initial cost of contract deployment. Bottom: we see results for each individual phase. Here, we see gas costs for each phase also increase linearly with c, apart from the calculation phase, where the operator calculates and publishes the auction results, which depends on the number of matches, m.

Duration also follows a similar linear relationship, apart from when $c = 10$, which is skewed by phase transitions. Overall, results for the duration of each individual phase, and therefore the whole trading period, deviate from the theoretical minima. This occurs because each block is not fully filled with transactions from the protocol, as the theoretical calculation assumed. One reason for this is interference from other people using the testnet. More significantly, experiments were performed using one computer to simulate all clients. Therefore, there was not enough time to generate enough transactions to fill each block. In reality, the trading and reveal phases will be performed by multiple clients in parallel, so duration will be significantly reduced. For the other phases, conditions such as network delays and processing speed on the operator's machine will be a major factor in protocol duration. Finally, the processing time for the calculation phase denotes the time taken for the operator

4. See Harmony Block Explorer results for 10, 100, and 1000 clients.

to validate orders and perform matching. This also grows linearly as the majority of time is spent validating orders rather than calculating auction outcomes.

5. Comparative Analysis

The intersection of blockchain technology with secure double auctions has not been extensively studied. Additionally, most research in this general area tends to utilise blockchain technology very differently to the protocol we present. One notable exception is the work of Galal and Youssef [19], who similarly use a commitment scheme in combination with periodic auctions. We therefore present a comparison between the key aspects of the two works. Henceforth, we will use P_{GY} to refer to the protocol in [19] and P_{BA} to refer to the Block Auction protocol introduced in this paper.

Broadly speaking, protocol P_{GY} works as follows.[5] *Setup*: First the operator deploys the contract and sets a number of parameters including their public key, the duration of each phase, and the number of commitments. The operator, also provides a security deposit as collateral. *Phase 1*: Each client commits to a bid (or ask) and provides zero-knowledge proofs that verify that their commitment is within a range. Within a specified time window, the clients then submit their orders along with a security deposit. The proofs are checked by the smart contract and if correct the orders are stored in the list of bids (or asks). *Phase 2*: Clients get the public key of the operator from the smart contract and use it to encrypt their orders. They then submit this ciphertext along with zero-knowledge proofs that the ciphertext matches their commitment. The contract verifies the proofs and emits events to notify the operator of the reveal. *Phase 3*: The operator collects the revealed orders from the smart contract and performs the matching. The operator then publishes the results along with a zero-knowledge proof of correctness. Finally, the smart contract checks the proofs and returns the collateral (the deposit) to the parties that acted according to protocol.

5.1. Privacy and Verifiability

Security deposits are used as collateral in both protocols to incentivise clients to reveal their orders.[6] This also makes it prohibitively costly for the operator to simulate fraudulent clients and only reveal orders which benefit them.

In terms of privacy, the guarantees of P_{GY} can be considered worse than P_{BA} during order submission. Submitting an order using P_{BA} reveals no information about the order. However, in P_{GY} (i) the intention to trade a particular asset and (ii) the order direction are revealed. This gives valuable information to observers regarding the supply and demand, and makes it easier for malicious parties, including the operator, to guess a particular client's trading intention.

The protocol verification guarantees of P_{GY} are perhaps better than P_{BA}, since every action in P_{GY} comes with a zero-knowledge proof that proves its correctness. In P_{BA}, no such proofs are provided, however correctness proofs are not necessary for order submission or order reveal as those actions can later be proven by other means. The main benefit of correctness proofs in P_{GY} is so that the clients can confirm the correct execution of the protocol at the end of the trading period. In P_{BA}, some guarantees are available to individual clients, but complete verification of execution is achieved through the cooperation of all clients.

5.2. Completeness

P_{GY} is a functional protocol that achieves all the aims of its authors. But, as presented, it is not a complete system. The operator and the clients still need to manually interact with the blockchain as no applications are provided to reduce their work load. Additionally, it is not clear how the protocol deals with permissions or how it should be reset between each iteration. This is understandable, as the authors were not aiming to produce a complete system, only a proof-of-concept. On the other hand, we present P_{BA} as a complete system that can easily be deployed in a real world setting with a few minor modifications.

5.3. Flexibility

P_{GY} incorporates the single clearing price periodic auction. To accommodate other clearing/matching algorithms, the protocol would require adjustment since the proof of correctness provided by the operator is a zero-knowledge proof that the sorting of the orders is correct. This proof works with the single price periodic auction clearing method but does not necessarily work with other clearing algorithms and most probably a different proof will be needed for a different algorithm. Also, as the proof is checked by the smart contract, changing the clearing algorithm will require the redeployment of the contract. In contrast, the smart contract of P_{BA} is agnostic of the clearing algorithm and can work with any clearing algorithm that fits the periodic auction model. In the current implementation for example, three different clearing algorithms are presented but more can be easily added. Additionally, in P_{GY} each trading period only works for a particular asset, while P_{BA} enables trading in any number of assets. These differences make P_{BA} more flexible than P_{GY}.

5.4. Execution Cost

As the two protocols have different functions, only a small subset of operations that perform a similar task can be

5. P_{GY} code: https://github.com/Anonymousub/PeriodicAuction

6. Looking at the smart contract of P_{GY}, we believe that the security deposits do not work as described in [19]. The paper states that deposits are refunded to unexecuted orders, and for executed orders deposits are locked in the contract until they are settled. Additionally, if the operator fails to clear the market all clients should be refunded. This functionality does not seem to be present in the contract code as the only places where deposits are returned are when disputes take place. More importantly, if the operator is allowed to receive a client's deposit by disputing their order, this can lead to an exploit where the operator creates fraudulent clients and submits multiple orders; then only those orders that benefit the operator are revealed, and other orders are disputed to receive their deposits.

directly compared. Deployment of the contracts seems to cost slightly less in P_{BA} than P_{GY}. However, this is of little concern as the protocols would ideally be deployed only once. In P_{GY}, the costs for submitting and revealing an order are presented as $276k$ and $49k$, respectively. According to the authors, these are theoretical approximations as they only take into account the cost for verifying the zero-knowledge proofs and additional operations are not taken into account. Running the provided demo shows the actual gas costs for those operations are $234k$ gas for submitting an order and $115k$ gas for revealing it. The same operations in P_{BA} cost $53k$ and $171k$ gas, respectively. As a client needs to both submit and reveal their order to participate in the auction, in P_{GY} the client pays $350k$ gas, while in P_{BA} the client pays $223k$ gas. Therefore, P_{BA} is more efficient for clients.

The market clearing function of P_{GY} and the matching publication function of P_{BA} are harder to compare as they work differently. According to [19], the clear market operation of P_{GY} can theoretically process ~ 728 orders using a $10M$ gas limit, which translates to $14k$ gas per order.[7] This does not take into account any costs apart from the costs for verification of the zero-knowledge proofs. In practice, using the open source demo provided, we were able to process only 106 orders before we reached the default gas limit of Ganache ($\sim 6.7M$ gas). This experimental result translates to a cost per order of around $63k$ gas. As each matching publication in P_{BA} costs either $95k$ or $149k$ gas, the operator using P_{BA} would have to pay $74.5k$ gas per matched order at best or $95k$ at worst. Note that the unmatched orders should not be counted in the calculation for P_{BA} as only matched orders are published, while in P_{GY} all orders contribute to the cost. Using the experimental results for both protocols, we can calculate that P_{GY} is more efficient than P_{BA} for auctions where the ratio of executed orders to all orders (REO) is greater than 0.66 (using $95k$) or 0.85 (using $74.5k$). In practice, we think that the REO of real-life auctions will rarely be that high, so P_{BA} will be more efficient most of the time. Additionally, P_{BA} appears to be more efficient for auctions with few orders as it takes a while for P_{GY} to converge to the $63k$ gas cost per order mentioned above.

5.5. Scalability

Scalability is important for any practical protocol. Again, due to differences in the way the two protocols work, it is difficult to make precise scalability comparisons. As mentioned before, it is not clear how P_{GY} deals with permissions, how it resets for the next iteration, or if it should be deployed again. This is a critical comparison, as a large number of transactions are needed in P_{BA} for those operations and as we cannot evaluate this number for P_{GY} those comparisons will not be discussed.

Scalability for clients depends on order submission and order reveal. For P_{GY}, each order submission costs $234k$ gas, so a maximum of 64 orders can be executed before the gas limit of a single block is reached (for P_{BA}, this number is 288). For P_{GY}, each order reveal costs $115k$ gas, so 130 reveals can fit in a block (for P_{BA} this number is 88). Therefore, the minimum duration, dur, of the order submission and order reveal phases can be calculated as a function of the number of clients, c, such that:

$$dur(P_{BA}) = \left\lceil \frac{c}{288} \right\rceil + \left\lceil \frac{c}{88} \right\rceil \leq dur(P_{GY}) = \left\lceil \frac{c}{64} \right\rceil + \left\lceil \frac{c}{130} \right\rceil$$

i.e., for any c, P_{BA}'s order submission and reveal are quicker.

The greatest differences lie in the actions of the operator. In P_{GY}, the operator only needs one transaction to clear the whole market, while in P_{BA} the operator publishes all matched orders and one transaction is needed for each. In this sense, P_{GY} is clearly more scalable. However, since one transaction clears the whole market, there is a maximum number of orders that P_{GY} can accommodate. The authors of P_{GY} state that this limit is ~ 728 orders (using $10M$ gas limit, which becomes ~ 1092 orders using $15M$ limit), but this value does not take into account any gas costs other than the costs for verification of proofs. In practice, with a $6.7M$ gas limit, we found this number to be 106. Adjusting for the current $15M$ gas limit gives P_{GY} an order limit of ~ 238. In comparison, P_{BA} has no such limit as each match is cleared individually. Therefore, while P_{BA} scales linearly with the number of orders, P_{GY} just requires a constant time of one for auctions with fewer than 238 orders and then cannot continue scaling. This is a hard limitation on P_{GY}.

5.6. Summary of Comparison

Both protocols have advantages and disadvantages in scalability, and execution cost, P_{GY} provides better verifiability while P_{BA} is superior in terms of privacy, flexibility and completeness. Regarding verifiability, P_{GY} provides ZK proofs of auction execution, while P_{BA} does not. Regarding privacy, P_{GY} reveals the asset and direction of an order, while P_{BA} does not. In terms of execution cost, P_{GY} is more efficient for the operator for auctions with $REO > 0.66$ (worst case) or $REO > 0.85$ (best case). For auctions with other REOs, and for the clients, P_{BA} is more cost efficient. P_{BA} provides better scalability on the number of clients, while P_{GY} provides constant scalability on the number of orders up to a limit (~ 238). After this point, P_{BA} can continue to scale while P_{GY} cannot. P_{BA} is more flexible as multiple auction clearing algorithms are interchangeable while P_{GY} only supports one auction mechanism that is hard-coded in the smart contract. P_{BA} allows the trade of multiple assets while P_{GY} can trade only one. Finally, P_{BA} is provided with a full implementation, including dedicated client and operator applications, making it a complete open source system ready for use. In contrast, only the smart contract and a demo is provided with P_{GY}.

7. We noticed a small bug in the code of the clear market function of P_{GY}. If the clearing price is a decimal number, for example 100.5, then 101 (or 100) should be accepted as the contract does not work with decimal numbers. But, from our experimentation, it seems that neither works, thus the operator cannot clear the market in such cases.

6. Conclusion

We have presented a novel protocol for privacy-preserving and verifiable periodic double auctions using blockchain technology, and released a full system implementation. The protocol is customisable to accommodate any periodic double auction mechanism and can be easily transformed into a commercial system. The protocol was evaluated both theoretically and experimentally, with results demonstrating clear advantages over the closest alternative protocol found in the literature. Having a privacy-preserving double auction with a publicly verifiable audit trail not only benefits dark pool financial trading, but offers public benefit in application areas such as government procurement.

Acknowledgments

This work was supported by Innovate UK / ESRC funded Knowledge Transfer Partnership between University of Bristol and Claritum Limited (KTP 11952). John Cartlidge is sponsored by Refinitiv.

References

[1] ESMA, "MIFID II," Policy Document, 2018. [Online]. Available: https://www.esma.europa.eu/policy-rules/mifid-ii-and-mifir

[2] J. Cartlidge, N. P. Smart, and Y. Talibi Alaoui, "Multi-Party Computation Mechanism for Anonymous Equity Block Trading: A Secure Implementation of Turquoise Plato Uncross," Cryptology ePrint Archive, Report 2020/662, 2020. [Online]. Available: https://eprint.iacr.org/2020/662

[3] R. Alvarez and M. Nojoumian, "Comprehensive survey on privacy-preserving protocols for sealed-bid auctions," *Computers & Security*, vol. 88, no. 101502, 2020. [Online]. Available: https://doi.org/10.1016/j.cose.2019.03.023

[4] P. Bogetoft, I. Damgård, T. Jakobsen, K. Nielsen, J. Pagter, and T. Toft, "A Practical Implementation of Secure Auctions Based on Multiparty Integer Computation," in *Financial Cryptography and Data Security*, 2006, pp. 142–147. [Online]. Available: https://doi.org/10.1007/11889663_10

[5] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. Schwartzbach, and T. Toft, "Secure Multiparty Computation Goes Live," in *Financial Cryptography and Data Security*, 2009, pp. 325–343. [Online]. Available: https://doi.org/10.1007/978-3-642-03549-4_20

[6] C. S. Jutla, "Upending Stock Market Structure Using Secure Multi-Party Computation," Cryptology ePrint Archive, Report 2015/550, 2015, https://eprint.iacr.org/2015/550.

[7] J. Cartlidge, N. P. Smart, and Y. Talibi Alaoui, "MPC Joins The Dark Side," in *ACM Asia Conference on Computer and Communications Security (Asia CCS)*, 2019, p. 148–159. [Online]. Available: https://doi.org/10.1145/3321705.3329809

[8] C. Thorpe and D. C. Parkes, "Cryptographic Securities Exchanges," in *Financial Cryptography and Data Security*, 2007, pp. 163–178. [Online]. Available: https://doi.org/10.1007/978-3-540-77366-5_16

[9] ——, "Cryptographic Combinatorial Securities Exchanges," in *Financial Cryptography and Data Security*, 2009, pp. 285–304. [Online]. Available: https://doi.org/10.1007/978-3-642-03549-4_18

[10] J. J. Sikorski, J. Haughton, and M. Kraft, "Blockchain technology in the chemical industry: Machine-to-machine electricity market," *Applied Energy*, vol. 195, pp. 234–246, 2017. [Online]. Available: https://doi.org/10.1016/j.apenergy.2017.03.039

[11] M. Foti and M. Vavalis, "Blockchain based uniform price double auctions for energy markets," *Applied Energy*, vol. 254, no. 113604, 2019. [Online]. Available: https://doi.org/10.1016/j.apenergy.2019.113604

[12] S. Thakur, B. P. Hayes, and J. G. Breslin, "Distributed Double Auction for Peer to Peer Energy Trade using Blockchains," in *5th Int. Symp. on Environment-Friendly Energies and Applications (EFEA)*, 2018, pp. 1–8. [Online]. Available: https://doi.org/10.1109/EFEA.2018.8617061

[13] T. D. T. Nguyen and M. T. Thai, "Trustless Framework for Iterative Double Auction Based on Blockchain," in *Security and Privacy in Communication Networks (SecureComm)*, 2019, pp. 3–22. [Online]. Available: https://doi.org/10.1007/978-3-030-37228-6_1

[14] E. Mengelkamp, J. Gärttner, K. Rock, S. Kessler, L. Orsini, and C. Weinhardt, "Designing microgrid energy markets: A case study: The Brooklyn Microgrid," *Applied Energy*, vol. 210, pp. 870–880, 2018. [Online]. Available: https://doi.org/10.1016/j.apenergy.2017.06.054

[15] T. Zhang and L. Wang, "Republic Protocol: A decentralized dark pool exchange providing atomic swaps for Ethereum-based assets and Bitcoin." 2017. [Online]. Available: https://republicprotocol.github.io/whitepaper/republic-whitepaper.pdf

[16] F. Massacci, C. N. Ngo, J. Nie, D. Venturi, and J. Williams, "FuturesMEX: Secure, Distributed Futures Market Exchange," in *IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 335–353. [Online]. Available: https://doi.org/10.1109/SP.2018.00028

[17] L. Liu, M. Du, and X. Ma, "Blockchain-Based Fair and Secure Electronic Double Auction Protocol," *IEEE Intelligent Systems*, vol. 35, no. 3, pp. 31–40, 2020. [Online]. Available: https://doi.org/10.1109/MIS.2020.2977896

[18] C. N. Ngo, F. Massacci, F. Kerschbaum, and J. Williams, "Practical Witness-Key-Agreement for Blockchain-based Dark Pools Financial Trading," in *Financial Cryptography and Data Security*, 2021. [Online]. Available: https://fc21.ifca.ai/papers/113.pdf

[19] H. S. Galal and A. M. Youssef, "Publicly Verifiable and Secrecy Preserving Periodic Auctions," in *5th Workshop on Trusted Smart Contracts (WTSC'21)*, 2021. [Online]. Available: https://fc21.ifca.ai/wtsc/WTSC21paper2.pdf

[20] J. Niu and S. Parsons, "Maximizing Matching in Double-sided Auctions," 2013. [Online]. Available: https://arxiv.org/abs/1304.3135

[21] A. T. Pollacco, "The Interaction between Blockchain Evidence and Courts – A cross-jurisdictional analysis," https://blockgeeks.com/guides/blockchain-evidence/, accessed: 19/07/2021.

[22] Harmony Team, "Harmony Technical Whitepaper." [Online]. Available: https://harmony.one/whitepaper.pdf

[23] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains," in *13th EuroSys Conference*, 2018. [Online]. Available: https://doi.org/10.1145/3190508.3190538

[24] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, 2014. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf

[25] V. Buterin, "On Settlement Finality," Ethereum Foundation Blog, 2016. [Online]. Available: https://blog.ethereum.org/2016/05/09/on-settlement-finality/