All for one and one for all: Fully decentralised privacy-preserving dark pool trading using multi-party computation

Mariana Botelho da Gama[1], John Cartlidge[2], Nigel P. Smart[1,2], and Younes Talibi Alaoui[1]

[1] imec-COSIC, KU Leuven, Leuven, Belgium.
[2] Department of Computer Science, University of Bristol, Bristol, UK.
mariana.botelhodagama@kuleuven.be,john.cartlidge@bristol.ac.uk, nigel.smart@kuleuven.be, younes.talibialaoui@kuleuven.be

Abstract. Financial dark pool trading venues are designed to keep pre-trade order information secret so that it cannot be misused by others. However, dark pools are vulnerable to an operator misusing the information in their system. Prior work has used MPC to tackle this problem by assuming that the dark pool is operated by a small set of two or three MPC parties. However, this raises the question of who plays the role of these operating parties and whether this scenario could be applied in the real world. In this work, we implement an MPC-based dark pool trading venue with up to 100 parties. This configuration would allow a real-world implementation where the operating parties are the active participants that trade in the venue (i.e., a "no operator" model), or where the parties are the main stakeholders of the venue (e.g., members of a non-profit partnership such as Plato). We use AWS cloud to empirically test the performance of the system. Results demonstrate that the system can achieve trading throughput required for some real-world venues, while the cost of hosting the system is negligible compared with the savings expected from guaranteeing no information leakage.

Table of Contents

1 Introduction

In January 2022, the Securities and Exchange Commission (SEC) ordered tZERO, a US-registered "dark pool" operator, to cease and desist from violating the Exchange Act and pay a civil money penalty of US $800,000 [Uni22]. For more than two years, tZERO had failed to inform the SEC, or its customers, that it was sharing "non-displayed" order information with third parties, including

Blue Ocean Technologies, a Singapore-based broker-dealer, which tZERO acquired during this period. The *raison d'être* of a dark pool is to hide the trading intention of market participants by not displaying unfilled order information, as this information can be used by other traders to adversely move the price. Therefore, by selling order information to third parties, tZERO actively worked against its own customers that it was purporting to protect. Disappointingly, this unethical behaviour is not uncommon, and tZERO merely becomes the latest member to join an infamous group of dark pool operators that have collectively paid hundreds of millions of dollars in penalty settlements to the SEC for similar violations over the last decade (for a detailed list of dark pool violations, see [CST21, p.242, Table 1]).

A dark pool is a financial trading venue where investors can buy and sell financial instruments, such as equities and derivatives, without revealing their trading intention. When an investor submits a buy or sell order into the dark pool the order is not disclosed for other market participants to view. Rather, the order will wait "in the dark" until it is either removed by the owner, or until a matching counterparty order is discovered, at which point a trade will execute between the buyer and the seller (the exact mechanism used for matching orders varies between venues). By not displaying orders, dark pools are designed to stop predatory trading practices such as front-running that use the trading intention disclosed in an order against the owner of that order (e.g., see [CST19]). However, if an operator decides to abuse their privileged access to order information, the intended benefits of the dark pool are lost. Therefore, trust is a widely acknowledged issue for dark pool operators.

To address the issue of trust in dark pools, one approach is to replace a single dark pool operator (the case $n = 1$) by a set of independent parties (i.e., $n > 1$) who emulate the single operator via a multi-party computation (MPC) protocol, such that internal algorithm data is processed in secret-shared form between the n parties. Then, as long as some given ratio of parties remain honest, the internal data cannot be accessed maliciously. In 2019 [CST19], this secure "multiple operators" architecture (with $n = 2$ and $n = 3$) was shown to be capable of handling trading throughput (in a single instrument) equivalent to that expected in some real-world financial dark pool venues. Subsequently, the multiple operators model (with $n = 2$ and $n = 3$) has been used to emulate an approximation of Turquoise Plato, Europe's largest dark pool trading venue, and shown to be capable of securely handling real-world trading throughput across thousands of instruments [CST21].

Together, the previous works leave little doubt that MPC is ready for real-world implementation of a secure financial dark pool. However, they leave some important questions unresolved. In particular: *who should play the role of the n parties that operate the dark pool?* In prior works, it was suggested that a regulator, a primary exchange, or a rotating collection of stakeholders (such as members of the Plato Partnership) could act as MPC parties; but none of these solutions fully address the issue and all have some drawbacks. Here, for the first time, we extend the previous MPC model to many parties $l \gg 3$, such that every stakeholder, for example every investor that trades in the dark pool, can act as an operating party in the MPC protocol (we switch to use notation l to indicate that parties can participate in the market, whereas n indicates that parties are non-participating operators). We examine the situation where the MPC-protocol is executed by a set of $l \leq 100$ stakeholders, which simulates the organisational structures of some real world dark pools. We describe these next.

Given that the purpose of a dark pool is to prevent information leakage, particularly for large "block" orders that are most vulnerable to adverse market impact, the majority of dark pool venues restrict access to select members. The classic example is Liquidnet, a private dark pool venue for large institutional investors (buy-side firms, such as pension funds, that have extremely large assets under management) to anonymously trade large blocks of stocks. By restricting access to "natural liquidity" only (i.e., investment managers making trading decisions based on long-term fundamental analysis, rather than "technical" traders using market signals for short-term profits), Liquidnet has successfully cornered the market in large scale trades.[3] While global membership of Liquidnet has grown to 900 over the last twenty years, initially there were fewer than 100 participating members. For dark pool venues of this nature, the number of participating stakeholders will always be relatively small. Other dark pool venues, such as Turquoise Plato, encourage greater liquidity flow by allowing many smaller investors to trade on the platform. Operated by the London Stock Exchange Group, Turquoise Plato is overseen by the not-for-profit Plato Partnership, which includes around 25 major buy-side and sell-side members that work together as key stakeholders to ensure market fairness.[4]

By extending the multiple operators MPC-protocol to cover $l \leq 100$ parties, in this work we are able to instantiate a secure emulation of both a *Liquidnet-style* model, a fully-decentralised "no operator" model where all $l = 100$ parties are active trading participants (i.e., the trading venue is available to members only and all members act as an MPC party), and a *Plato-style* model, where the $l = 25$ operating parties are key stakeholders of the venue (i.e., the MPC parties are the largest investors on the platform, with legal oversight of the venue).

Contribution: We show that the volume matching algorithm used in some prior work, e.g., [CST19], can be realistically deployed for up to $l = 100$ parties; obtaining not only an acceptable throughput of orders but also at a sub-dollar cost per auction. These results demonstrate real-world commercial viability. Our experiments are performed, just as in [CST19], in the case of a universe containing a single stock; however extension to multiple stocks can be accomplished in a standard manner using the method in [CST21]. An overview of our results, in comparison to prior work on secure dark pools, is presented in Table 1.

2 Background

2.1 Related work

Here, we summarise recent related work that is most pertinent to this study. We categorise this work into blockchain-based frameworks, where solutions are designed to tackle the inherent challenges of trading securely over a distributed ledger, and work that is designed to secure dark pool trading systems in more traditional financial settings. In this work, we specifically address the latter problem. We do not attempt to solve the problem for blockchain-based systems, although future work could explore extensions in this area. For a more general review of privacy preserving auctions and the use of MPC, see [CST21].

Blockchain-based "dark pool" architectures Several recent works have introduced decentralised blockchain-based architectures designed to preserve elements of privacy in a variety of trading settings, including a futures exchange [MNN+18], an over-the-counter (OTC) market [NMKW21],

[3] In 2022, Liquidnet report \$79bn average daily liquidity and \$1.5m average execution size. See: https://www.liquidnet.com/equities-trading-solutions

[4] See: https://platopartnership.com/

Table 1. Performance comparison of dark pool architectures using volume matching for traditional (i.e. non-blockchain) markets. Time taken to match 1000 orders: number of operating parties (n) in prior works or participating parties (l) in this work; tolerance of number of adversarial parties (t); actively secure protocol (a); volume match protocol secures buy/sell direction (d); volume match includes Minimum Execution Size (MES); protocol (FT: Full Threshold (SPDZ) or HM: Shamir-Based Honest Majority or FHE: FHE-Based); online runtime in seconds (online); offline time as a multiple of online time (offline); universe of stocks (U).

	n/l	t	a	d	MES	MPC Protocol	Online	Offline Multiple	U
DarkSide[CST19]	2	1	✓	✗	✗	FT	0.5	450	1
DarkSide [CST19]	3	1	✓	✗	✗	HM	0.9	4.5	1
Plato [CST21]	2	1	✓	✗	✓	FT	¡ 5	150-350	4500
Plato [CST21]	3	1	✓	✗	✓	HM	¡ 5	3-8	4500
Bucket [dGCP$^+$21]	3	1	✓	✓	✗	HM	0.36	Not reported	1
SecretMatch [BDP20]	1	0	✗	✓	✗	FHE	800	0	1
Our Work	20	19	✓	✗	✗	FT	4.5	550	1
Our Work	20	1	✓	✗	✗	HM	3.0	2	1
Our Work	20	9	✓	✗	✗	HM	8.5	1.3	1
Our Work	100	99	✓	✗	✗	FT	20	?	1
Our Work	100	1	✓	✗	✗	HM	12	2.3	1
Our Work	100	49	✓	✗	✗	HM	133	1.4	1

a sealed-bid auction [BHSR20], and privacy-preserving decentralised exchanges [BDF21,GVJR21]. We briefly describe these important contributions, however we note that blockchain trading architectures introduce challenges that are subtly different to the challenges that must be solved when securing traditional financial dark pools. In particular, blockchain-based trading architectures must stop third parties from linking trades with traders, and must prevent front-running strategies that exploit public knowledge of pending transactions on the network and the miners' ability to determine the final transaction ordering (for systematic treatments of these issues, see [ByCD$^+$21,HW22]).

In [MN21], Massaci and Ngo detail the design principles necessary to secure blockchain-based distributed financial exchanges. They highlight the critical importance of anonymity and confidentiality when trading on a public blockchain, as attributes of an actor are recorded on-chain and can be tracked and used against them for discriminatory practices. In the scenario of a decentralised futures market, the authors describe one such attack scenario, where a trader is identified as having limited cash reserves and then forced by adversaries to liquidate an otherwise profitable position at a loss [MNN$^+$17,MN21]. To counter this very problem, in 2018, Ngo et al.[MNN$^+$18] introduced a decentralised futures market, where traders hide behind a Tor network to communicate anonymously and MPC is used to enable privacy of who is trading. However, as the order book is displayed, this architecture is not applicable for dark pool trading [Ngo19, p.89].

In 2021, Ngo et al [NMKW21] introduced an over-the-counter (OTC) trading protocol for counterparties to communicate and negotiate anonymously and privately over a permissionless blockchain acting as a public bulletin board. Hiding behind a Tor network to protect anonymity, counterparties broadcast price ranges that they are prepared to trade, but do not reveal their asset holdings. Bilateral negotiation proceeds as a series of iterative steps where each counterparty publicly commits to information that meets some desired relation (i.e., price range and cash capacity), until a trade agreement is reached.

In 2020, Bag et al. introduced a publicly verifiable decentralised protocol for performing a sealed-bid auction with no auctioneer. Using a public bulletin board (i.e., a permissionless blockchain) and an anonymous veto protocol, bidders are able to jointly compute the maximum bid while preserving the privacy of losing bids. The auction winner is able to prove they are the winner and other parties can check that there is only one winner or if there is a tie. This protocol cleverly solves the problem for single-sided auctions, where bidders are competing for one item, however it is not obvious how this protocol could be extended to a double auction, which is a necessary mechanism for a dark pool trading venue.

In 2021, two independent teams introduced privacy-preserving protocols for a decentralised exchange [BDF21,GVJR21]. The primary aim of these works is to ensure that exchange transactions are performed correctly. Both protocols incorporate a number of similar features, including periodic auctions that match limit orders of unit volume on price only, a small number of MPC parties performing sorting and/or matching off-chain, and a smart contract to record order commitments and transactions. The protocol of [BDF21] is designed for exchanging assets over multiple public ledgers and the authors suggest that the outsourced MPC servers that perform matching will be run by a small number of public organisations. In comparison, [GVJR21] suggest that a small number of brokers will act as MPC parties to first sort orders, before a smart contract implemented on a permissioned ledger (such as Hyperledger Fabric) obliviously performs the matching. However, these differences between [BDF21] and [GVJR21] are relatively minor compared with the similarity in architectures.

Dark pool solutions for "traditional" finance Two architectures have been proposed to tackle the problem of securing dark pools in a traditional financial setting. First, there is the "multiple operator" model, where internal algorithm data is held in secret-shared form and processed by a set of n independent servers, such that data cannot be accessed maliciously as long as a given ratio of servers remain honest. The second approach is the "single operator" model, where orders are sent in encrypted form and the operator uses properties of homomorphic encryption to match the encrypted orders obliviously. We consider both of these methods, below:

Multiple operating parties: The seminal work of Bogetoft et al [BDJ+06,BCD+09] introduced the first practically feasible MPC protocol for performing a secure double-auction using multiple operators. Traders first submit encrypted strategies that specify the quantity they are willing to buy or sell at each possible price point. The operating parties then securely calculate a single market clearing price that best balances aggregate supply with aggregate demand. In January 2008, the protocol was successfully deployed to run the Danish sugar beet auction, where farmers trade contracts for production on a nationwide-market [BCD+09]. The three operating parties included Danisco (the only sugar beets processor in Denmark), DKS (the sugar beet growers' association), and the research team who designed the protocol. Making the assumption that the operators would not maliciously deviate from the protocol, Shamir secret sharing with semi-honest (i.e., passive) security was used. A total of 1229 farmers entered bidding strategies across 4000 potential price points and the clearing price calculation was performed during one calendar day.

More than a decade after Bogetoft et al's groundbreaking work Cartlidge et al [CST19] demonstrated for the first time that MPC, with $n = 2$ and $n = 3$ operators, can emulate a secure dark pool trading venue. Protocols were introduced for three auction mechanisms commonly used in financial markets: (i) a *continuous double auction* (CDA), where buyers and sellers can post bids and offers at any time and a limit order book is used to perform continuous matching; (ii) a *periodic* *double auction*, where buyers and sellers first submit bids and offers during an open auction period before a single clearing price is calculated for all matches; and (iii) a *periodic volume match*, where orders submitted during the open period contain a value for quantity only (i.e., orders contain no limit price) and all matches trade at a single price determined by some external reference value (e.g., the current mid-price on the primary exchange). For trading in a single stock/instrument, empirical throughput results (number of orders submitted per second) showed that the CDA could handle 50-150 orders/s for small order books; the period double auction could handle 500 orders/s; and the periodic volume match could process between 1000 ($n = 3$ parties) and 2000 ($n = 2$ parties) orders/s. These throughputs are within the performance range required for many real-world financial dark pool venues. The volume match protocol is presented in Algorithm 1, which is the matching algorithm which we will focus on in this paper.

In 2021, Cartlidge et al [CST21] extended the 2-operators and 3-operators model to emulate a secure approximation of Turquoise Plato, Europe's largest dark pool by trading volume. Secure protocols implemented an approximation of Turquoise Plato Uncross, where order submissions contain a minimum execution size (MES) and a periodic double auction matches orders every 5 seconds. In addition, a *gateway allocation* protocol was introduced to enable secure trading across multiple stocks for the first time. Empirical results demonstrated that the system could handle trading across the full universe of 4,500 stocks/instruments with throughput equivalent to the real world trading venue.

Recently, in 2022 [dGCP$^+$21], more efficient volume matching protocols were introduced for dark pools with 2-operators and 3-operators. These extensions included methods to secure the *direction* of an order (whether it is an order to buy or sell), the introduction of dummy orders, and an exploration of fixed-size *bucket* matching. Empirical results demonstrated a three-fold improvement in throughput and less information leakage than the original volume matching protocol introduced in [CST19].

Single operator: Recently, an alternative single-operator architecture that makes use of Fully Homomorphic Encryption (FHE) has been proposed by a research team at JP Morgan. In this framework, clients send orders to the operator in encrypted form and the operator uses homomorphic properties of the encryption scheme to obliviously discover matches on the encrypted order data. Matched orders are then decrypted and executed. This architecture was first described at a high level, without a concrete implementation, in both a 2019 patent filing [ABPV21] and an extended abstract published in summer 2020 [ABPV20]. Neither of these works describe how matching can be computed. The first proof-of-concept instantiation of the architecture was presented at the end of 2020 [BDP20]. Named "SecretMatch", the protocols in [BDP20] are designed to tackle the problem of inventory matching, where a sell-side firm, such as a broker or bank, internally matches the buy and sell order flow of their clients, without sending their respective orders to a public exchange. The SecretMatch protocols instantiate a periodic volume match algorithm (similar to that presented in [CST19]) for a single symbol/instrument. Runtimes demonstrate that the system can process 1024 orders in just over 800 seconds. However, the authors admit that these results are optimistic as the system omits several features. Importantly, they omit the threshold key-generation and threshold decryption algorithms and instead use a vanilla FHE scheme with a single FHE decryption key [BDP20].

Runtime Comparison: Table 1 presents an approximate runtime comparison for processing 1000 orders using some variation of the volume matching algorithm (where orders are matched on volume only and transaction price is taken from some external reference value). We present not

Table 2. Summary of critiques of dark pool with multiple operating parties and brief response. Column A indicates critique is addressed in this work; column B indicates critique applies only to blockchain-based implementations.

Section	Critique	References	Response	A	B
2.2	No trader anonymity (on public ledger)	[Ngo19, p.89], [MN21, p.62], [NMKW21, p.583]	Only applies to blockchain implementations	✗	✓
2.2	Security with abort: cannot just "walk away"	[MN21, p.62], [BDF21, p.168]	Traditional markets can absorb a "halt" in trading	✗	✓
2.2	Regulator cannot act as an operator	[Ngo19, p.89], [ABPV20, p.1748], [BDP20, p.3], [MN21, p.60]	In our work, parties can include any major stakeholder	✓	✗
2.2	Operating parties may collude	[ABPV20, p.1748], [BHSR20, p.2051]	Stakeholders are incentivised to not collude	✓	✗
2.2	Can it scale to many parties?	[Ngo19, p.89], [MN21, p.69]	We demonstrate scaling to 100 stakeholders	✓	✗
2.2	Economic burden on operating parties	[NMKW21, Appendix p.21]	We show the cost of participating in MPC is very low	✓	✗
2.2	Heavy pre-processing required	[BDP20, p.3]	Parallelised and just-in-time pre-processing is possible	✗	✗
2.2	Leaks buy/sell direction	[ABPV20, p.1748], [BDP20, p.3]	A simple solution to secure direction is presented in [dGCP+21]	✗	✗

only the run-times from prior works, but also, for comparison, the results we obtain. It can be seen that the single operator architecture of [BDP20] is three orders of magnitude slower than the multiple operators architecture, despite [BDP20] using a trivial security threshold that significantly reduces computation time. Unlike the single operator architecture, which has negligible offline processing times, the multiple operators architecture [CST19,CST21,dGCP+21] requires significant offline processing that can be hundreds of times longer than online processing times. However, offline processing can be trivially parallelised, so these times are not a limiting factor on runtimes, but do incur additional server costs.

2.2 Critique of multiple operating parties

There have been multiple critiques of the MPC dark pool model with multiple operating parties.[5] A summary these critiques are presented in Table 2. We examine these criticisms below, but, in summary, one sees that the problems highlighted are either specifically related to the case of blockchain based dark pools, or are exactly related to the problem we address in this paper. We summarize the issues in Table 2.

Trader anonymity A recurring critique of the MPC dark pool is that the protocols do not preserve trader anonymity [Ngo19,MN21,NMKW21]. The authors of these works consider the related problem of dark pool trading over the public blockchain, where trading activity and trader information is publicly exposed. In particular, they detail an example attack in a blockchain futures market [MNN+17,MN21]. However, this attack relies on knowing a trader's exposure, which is possible in the blockchain space, but is easily avoided in a traditional financial market. In the MPC dark

[5] In March 2022, we performed an exhaustive search of all Google Scholar citations of [CST19,CST21,dGCP+21]. The majority of citations, and all critiques, are for the earliest work [CST19].

pool model, traders each have an account id that is encrypted in secret shared form and so is not exposed to the operator. Traders are also able, and likely, to simultaneously trade across multiple venues. Therefore, even if the operator is able to track an individual trader's executions, it is not possible to deduce a trader's full position. In addition, traders can operate through a broker, which obscures the communication origin. Therefore, while the authors are correct to raise anonymity as a concern for blockchain-based dark pool implementations, it is not relevant to our "traditional" financial application domain.

Security with abort The MPC dark pool protocols are in the active-with-abort security model, i.e., the protocols will abort if any party deviates from the protocol, as long as a threshold t of parties remain honest. It has been raised that it is not possible to simply "walk away" when malicious activity is detected [BDF21,MN21]. In particular a single dishonest party can ensure that the auction does not complete, i.e. the protocols are not robust. In the context of blockchain-based markets, e.g., see [ByCD$^+$21,HW22], this is indeed a problem. However, in traditional financial markets, it is possible (and not uncommon) to halt an exchange on detection of abnormal behaviour. Many markets now include "circuit breaker" functionality that halt trading in an asset during times of high volatility (e.g., when the asset price has moved more than 10% during one day). When the protocol aborts, the market will halt. In addition there is a form of self-interest in the stake-holders in a traditional market, of the stake-holders not wanting the market to be closed to themselves going forward.

Who acts as operator(s)? In [CST19], it was suggested that one of the operating parties could be played by the regulator. This comment has received criticism from both the blockchain domain [Ngo19,MN21] and the domain of traditional finance [ABPV20,BDP20]. In particular, it has been raised that regulators: i) are not allowed to actively participate in financial transactions; ii) do not have the computing capacity to act as an operator; and iii) the cost of a regulator performing this role will necessitate higher taxes. Although, to counter these one could state that i) a regulator's role can change over time; ii) an operator can make use of virtual cloud-hosted compute infrastructure which requires only a credit card to access; or iii) one would expect the cost to the regulator to be covered by the other commercial operators, which can be recovered through increased trading fees. However, we feel that this is indeed a major issue in prior works in the space of MPC-based dark pools. Thus, in our work, we extend the number of MPC operating parties by to up to 100. As we explained earlier, this matches to real world scenarios such as Liquidnet, where there is a natural number of less than 100 stake-holders members, or Turquoise Plato, where the number is in the range 15-20.

Operators may collude It has been noted that operating parties can collude to trivially access the secure information [ABPV20,BHSR20]. However, to achieve this, parties will need to openly plan to deceive their customers. Historically, a single operator can trivially cheat and remain undetected. If parties are chosen as key stakeholders, perhaps on a rotating basis, then there is incentive to not cheat as discovery of malicious activity will disallow future participation. In this paper, we demonstrate that the model scales to include key stakeholders, including traders, as participating parties.

Can it scale? Related to the prior point is the question as to whether the MPC-based methods can scale, which is raised in [MN21,Ngo19]. In this paper, we explore how the models scale to many more than $l = 3$ participating parties. We consider computation, communication and economic costs.

Economic burden on operators The provision of trading venues is a commercial activity that generates profit, therefore we expect operating parties to profit from trading fees. In [NMKW21] it is argued that running the MPC dark pool could be prohibitive. In this paper, we explore the economic costs of running an MPC dark pool for each participating party. We show that the costs are negligible, totalling less than one dollar per auction. Given each auction in a traditional market may trade millions of dollars worth of orders, the cost of one dollar to execute the auction is minimal in comparison and can easily be absorbed in fees; without leading to a significant loss in revenue for the market operator(s).

Heavy pre-processing Related to the economic costs is the fact that the prior MPC-protocols work in the pre-processing model of MPC. This is done in order to ensure that the final (online) trading can be executed as quickly as possible. However, the draw-back is that the underlying MPC protocols rely on heavy offline pre-processing that might make it infeasible at scale. In addition, the pre-processing requires one to know ahead of time the maximum market throughput [BDP20]. We note that, if necessary, pre-processing can be easily parallelised using scale-out across hundreds of elastic compute instances. Also, pre-processing can be performed close to just-in-time, as we demonstrate for the Shamir-based solutions in this paper. Using a combination of pre-processing performed at night, along with some just-in-time pre-processing, we see that if markets start to experience much higher throughput than the anticipated peak, additional processing can be performed.

Buy/sell direction is leaked The protocols in [CST19,CST21] make use of separate buy and sell order types. Therefore, the buy/sell direction of an order can leak. However, this can trivially be solved by having a single order type, where direction is a single bit parameter that is secret-shared. An example of this approach is introduced in [dGCP$^+$21]. Also, it is possible for a trader to simultaneously submit pairs of orders (one buy and one sell), where one order has zero volume. This leads to inefficiencies of additional order flow of dummy orders, but prevents leakage.

2.3 Dark pool with no operator

In this paper, we propose the "no operator" dark pool as an extension of the multiple operating parties model. Rather than have a small number of operators that provide the secure dark pool venue, the no operator model distributes the MPC dark pool computation between (some subset of) the market participants, i.e., by potentially including the traders and other stakeholders as MPC parties. This is in comparison to other works which simply consider "splitting" the dark pool operator into different entities. Figure 1 presents a schematic example. Similar to the multiple operating (split) parties model, in our model multiple MPC engines provide the secure functionality of the dark pool venue. However, these engines are owned by market participants who can trade in the venue on their own behalf. This model addresses the criticisms of *who acts as the operator?* (Section 2.2) and the charge of being susceptible to *operator collusion* (Section 2.2), as MPC parties

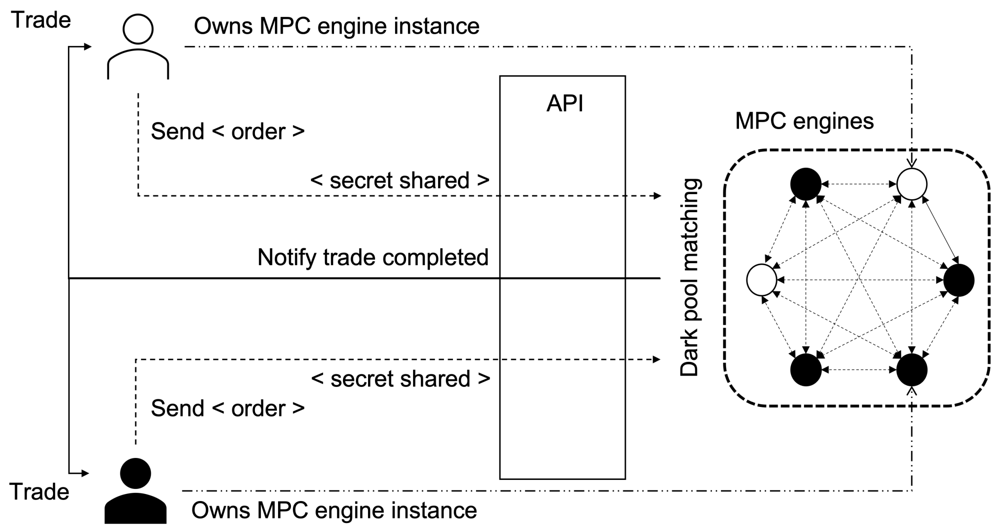

Fig. 1. Darkpool with no operator. The MPC engines that perform secure matching are owned by market participants.

are now active stakeholders in the market and therefore have strong incentive to not be excluded from the venue.

In the following sections, we empirically evaluate a concrete example of the model using AWS instances to instantiate each MPC engine. This enables us to accurately evaluate the economic costs of each participating party and we will show that costs are relatively small, therefore addressing the criticism of *economic burden* (Section 2.2). We also address the charge of *scaling* (Section 2.2) by showing that the system can scale to 100 MPC parties and still have much quicker throughput than the single operator model (see Table 1), while also offering active security with much greater tolerance to adversarial parties.

To enable comparison with previous work, we implement periodic volume matching (see Algorithm 1) using the original protocol presented in [CST19]. As shown in Table 1, this protocol is *not* the most efficient volume match implementation, so runtimes can trivially be improved by swapping to the more efficient volume match protocol presented in [dGCP+21].

Algorithm 1 Volume Match. First presented in [CST19].

This describes the algorithm to secure match N buy orders against M sell orders, in the time priority of the order number. Each buy (resp. sell) order is presented as the volume that is wished to be traded. The notation $\langle x\rangle$ denotes that the value x is stored within the MPC-engine in secret shared form, i.e. the value being operated on can be considered as "secret". That the operations, such as $\langle f\rangle \leftarrow \langle S\rangle > \langle B\rangle$, can be efficiently carried securely follows from standard work in the MPC literature.

Require: Number of sell orders N and number of buy orders M are known.
Input: Sell orders $\langle s_1\rangle, \dots, \langle s_N\rangle$ and buy orders $\langle b_1\rangle, \dots, \langle b_M\rangle$.
Output: The opened buy and sell orders, giving how much of each order was completed, with zero indicating none of order was executed.

```
 1: $\langle S\rangle \leftarrow \sum_{i=1}^{N} \langle s_i\rangle$                    ▷ $\langle S\rangle$ holds total sell volume
 2: $\langle B\rangle \leftarrow \sum_{j=1}^{M} \langle b_j\rangle$                    ▷ $\langle B\rangle$ holds total buy volume
 3: $\langle f\rangle \leftarrow \langle S\rangle > \langle B\rangle$                  ▷ $\langle f\rangle = 1$ if sell volume ¿ buy volume, else $\langle f\rangle = 0$
 4: $\langle T\rangle \leftarrow \langle f\rangle \cdot (\langle B\rangle - \langle S\rangle) + \langle S\rangle$    ▷ $\langle T\rangle$ holds total transaction volume
 5: $\langle L\rangle \leftarrow \langle T\rangle$                                ▷ $\langle L\rangle$ holds total volume left to transact
 6: for $i \leftarrow 1$ to $N$ do                      ▷ Calculate volume to transact for each sell order
 7:     $\langle z_1\rangle \leftarrow \langle L\rangle \leq 0$                  ▷ $\langle z_1\rangle = 1$ if no more volume to transact, else 0
 8:     $\langle z_2\rangle \leftarrow \langle L\rangle < \langle s_i\rangle$              ▷ $\langle z_2\rangle = 1$ if $i$ will not fully transact, else 0
 9:     $\langle s_i\rangle \leftarrow (((\langle L\rangle - \langle s_i\rangle) \cdot \langle z_2\rangle + \langle s_i\rangle) \cdot (1 - \langle z_1\rangle))$   ▷ Volume of sell order $i$ that will transact
10:     $\langle L\rangle \leftarrow \langle L\rangle - \langle s_i\rangle$                ▷ Reduce volume left to transact
11: end for
12: $\langle L\rangle \leftarrow \langle T\rangle$                                ▷ Reset total volume left to transact
13: for $j \leftarrow 1$ to $M$ do                      ▷ Calculate volume to transact for each buy order
14:     $\langle z_1\rangle \leftarrow \langle L\rangle \leq 0$
15:     $\langle z_2\rangle \leftarrow \langle L\rangle < \langle b_j\rangle$
16:     $\langle b_j\rangle \leftarrow (((\langle L\rangle - \langle b_j\rangle) \cdot \langle z_2\rangle + \langle b_j\rangle) \cdot (1 - \langle z_1\rangle))$   ▷ Volume of buy order $j$ that will transact
17:     $\langle L\rangle \leftarrow \langle L\rangle - \langle b_j\rangle$
18: end for
19: Open $\langle s_i\rangle$ and $\langle b_j\rangle$ for all $i$ and $j$             ▷ Open volume that each order transacts
```

3 MPC Background

We provide here the necessary background for MPC and the protocols we used. For more details we refer to the extensive MPC literature related to the protocols we use. In particular, the reader should refer to [Cd10,DPSZ12,DKL+13,CDI05,SW19,KRSW18].

3.1 MPC

MPC allows a set of distrustful parties to calculate a function f on their respective inputs, without having to reveal them. The parties in theory need to express the function f in the form of a circuit composed of additions and multiplications over some given arithmetic domain; however in practice there are various sub-protocols which deviate from this circuit representation in order to obtain improved performance for specific operations; for example the protocols in [Cd10] for performing basic arithmetic operations, such as comparisons, on integers. For simplicity we will focus more on a circuit model of computation in this overview, but the reader should be aware that in practice this is not strictly true.

In an MPC protocol, one needs to specify the capabilities of the adversary, that is, whether the adversary is willing to stick to the specification of the protocol, and only try to infer information about the secrets, from their view of the protocol, in which case we talk about a *passive adversary*, or that the adversary is willing to deviate from the protocol, in which case it is called an *active adversary.* One also needs to specify which security guarantees the protocol satisfies. For instance, whether it provides robustness, in which case it is guaranteed that the honest parties will obtain at the end of the protocol the output, or the adversary can prevent the honest parties from obtaining it.

In this work we shall focus on protocols which guarantee active-with-abort security. That is, either the protocol succeeds and all the parties receive the output, or the protocol aborts and the honest parties will learn this (i.e. that one of the parties deviated from the protocol). The base MPC functionality that realizes the protocols we are using is given in Figure 2.

MPC functionality $\mathcal{F}[\mathsf{MPC}]$

The functionality runs with $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_l\}$ and an ideal adversary $\mathcal{A}$, that statically corrupts a set of parties A. Given a set I of valid identifiers, all values are stored in the form (varid, x), where $\mathit{varid} \in I$.

Initialize: On input (init, p) from all parties, where p is a prime number, the functionality stores (domain, p),
Input: On input $(\mathit{input}, \mathcal{P}_i, \mathit{varid}, x)$ from party $\mathcal{P}_i$ and $(\mathit{input}, \mathcal{P}_i, \mathit{varid}, ?)$ from all other parties, with varid a fresh identifier and $x \in \mathbb{F}_p$, the functionality stores (varid, x) in memory.
Add: On input $(\mathit{add}, \mathit{varid}_1, \mathit{varid}_2, \mathit{varid}_3)$ from all parties, where $\mathit{varid}_1, \mathit{varid}_2$ are present in memory and varid_3 is not, the functionality retrieves (varid_1, x), (varid_2, y) and stores $(\mathit{varid}_3, x + y)$ in memory.
Multiply: On input $(\mathit{multiply}, \mathit{varid}_1, \mathit{varid}_2, \mathit{varid}_3)$ from all parties, where $\mathit{varid}_1, \mathit{varid}_2$ are present in memory and varid_3 is not, the functionality retrieves (varid_1, x), (varid_2, y) and stores $(\mathit{varid}_3, x \cdot y)$ in memory.
Output: On input $(\mathit{output}, \mathit{varid}, i)$ from all honest parties, where varid is present in memory, the functionality retrieves (varid, y) and outputs it to the environment. The functionality waits for an input from the environment. If this input is $\mathsf{Deliver}$ then y is output to all parties if $i = 0$, or y is output to party i if $i \neq 0$. If the adversarial input is not equal to $\mathsf{Deliver}$ then $\varnothing$ is output to all parties.

Figure 2. MPC functionality $\mathcal{F}[\mathsf{MPC}]$

3.2 Access structures

In order to choose a specific MPC protocol one needs to define what access (resp. adversary) structure that one is willing to support, i.e. from what set of players will the honest (resp. adversarial) parties be chosen. To fix notation we let $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_l\}$ be a set of l players and $2^{\mathcal{P}}$ be the corresponding power set. For $\Gamma, \Sigma \in 2^{\mathcal{P}}$, the tuple (Γ, Σ) is called an access structure if $\Gamma \cap \Sigma = \emptyset$, and the set Γ is monotonic (i.e. $A \in \Gamma$ implies all supersets of A are also in Γ, and $A \in \Sigma$ implies all subsets of A are also in Σ). The sets in Γ are called qualified sets and the sets in Σ are called unqualified sets. When $\Gamma = 2^{\mathcal{P}} \setminus \Sigma$, the access structure is called complete.

A typical example of a complete monotone access structure is a threshold access structure, where $\Sigma = \{S \subseteq 2^{\mathcal{P}}$ such that $|S| \leq t\}$ and $\Gamma = \{S \subseteq 2^{\mathcal{P}}$ such that $|S| > t\}$ for some threshold t. An access structure is called $\mathcal{Q}_2$ if no combination of two sets from Σ can cover $\mathcal{P}$. For instance, a threshold access structure with threshold $t < l/2$ is $\mathcal{Q}_2$

A set S in Σ is called maximally unqualified if there is no set S' in Σ such that $S \subset S'$. We will denote by $\mathcal{M}$ the set of maximally unqualified sets, and by Δ the set that contains the complements of maximally unqualified sets, i.e., $\Delta = \{\mathcal{P} \setminus S$ such that $S \in \mathcal{M}\}$.

3.3 Linear Secret Sharing Schemes

A Secret Sharing Scheme is defined over a given algebraic domain, for example in our case we take a finite field $\mathbb{F}_p$. This means that the values being shared will be elements $s \in \mathbb{F}_p$, and the data (the share) which is given to each party $\mathcal{P}_i$ will be a vector $\mathbf{s}_i$ of elements in $\mathbb{F}_p$. A Secret Sharing Scheme is a cryptographic mechanism that realises a specific access structure (Γ, Σ). That is, it allows one to secret share values such that any qualified set should be able to reconstruct them, and unqualified sets should not be able to do so. We write $\langle s \rangle$ to denote that the value s is shared under the given secret sharing scheme. A Secret Sharing Scheme is called Linear (LSSS for short) if, informally speaking, the parties can perform locally linear operations. That is, adding secrets and multiplying them by scalars. The two examples of LSSS which we will use in this paper are:

Shamir secret sharing which can realise a threshold access structure. In Shamir secret sharing, a secret $s \in \mathbb{F}_p$ is encoded as the constant term of a polynomial $f_s \in \mathbb{F}_p[X]$ of degree t with coefficients in a prime field $\mathbb{F}_p$. The share of player i consists of the evaluation of the polynomial f_s at the point $i \in \mathbb{F}_p$, i.e. $s_i = f_s(i)$. Reconstructing the secret s requires $t+1$ shares, thus only qualified sets recover s as they are the only sets with sizes bigger than t. Parties can calculate the sum of two secrets s and s' by just adding up locally their shares of s and s', as $f_s(0) + f_{s'}(0) = f_{s+s'}(0)$.

Additive secret sharing which consists of additively secret sharing a secret $s \in \mathbb{F}_p$, by giving every player a share $s_i \in \mathbb{F}_p$ subject to $s = s_1 + \ldots + s_l$. This LSSS realises a threshold access structure with threshold $t = l - 1$. That is, we need the l shares to recover the secret, each of which is held by only one player. Parties can calculate the sum of two secrets s and s', by just locally adding up their shares of s and s', as $s + s' = \sum_{i=1}^{i=l}(s_i + s'_i)$.

An LSSS is called multiplicative if the parties can obtain an additive sharing of the product of two secrets s and s' by just performing local computation. This is the case for Shamir secret sharing when $t < l/2$ as the product of two shares is a share of the product of the two secrets, but with a polynomial of degree $2 \cdot t < l$. Thus the sum of these new shares, each of which multiplied with a corresponding Lagrange coefficient, results in $s \cdot s'$. The additive secret sharing is not multiplicative, as the terms of the cross product cannot be calculated locally.

There is another form of secret sharing scheme which will be implicitly used, and which will explain some of our experimental results, namely the replicated secret sharing scheme:

Replicated secret sharing where a secret $s \in \mathbb{F}_p$ is shared among the parties by assigning a share to every set in Δ of the corresponding access structure. That is, the parties of every set D in Δ obtain a share s_D, subject to $s = \sum_{D \in \Delta} s_D$. Clearly, no unqualified set can reconstruct s, as it is missing the share that was assigned to its complement set. Qualified sets are also able to reconstruct the secret, as every set in Δ should contain at least one of its members. That is, if there exists a set in Δ where this is not true, then all those members belong to an unqualified set which contradicts the fact that these belong to a qualified set. It is straightforward why this secret sharing is linear, as we have that for another secret s', $s + s' = \sum_{D \in \Delta}(s_D + s'_D)$.

3.4 MPC-Protocols

As remarked above we shall utilize MPC protocols which are based on arithmetic in a finite field $\mathbb{F}_p$, and which provide active-with-abort security. We work in a pre-processing model, so that the online times are as efficient as possible. In the case of Shamir-based protocols we use a variant of the Smart–Wood protocol [SW19,KRSW18], which was shown in [JSL21] to provide the most efficient online phase in this situation. If one is interested in reducing the time for both online and offline phases the protocol of [CGH+18] is to be preferred; see [JSL21] for a complete comparison. In the case of full threshold based protocols we make use of the SPDZ protocol [DPSZ12,DKL+13].

All protocols offer active-with-abort security with a given cheating probability. The probability that a party cheats without being caught is one over the size p of the field. Thus taking p big enough (e.g. $log_2(p) = 128$) as we did for the experiments ensures that this probability is negligible. Note that this is a weaker security guarantee than robustness, however this security level is sufficient in many real world applications, and it allows to have efficient protocols so as to satisfy real world requirements. As we argued earlier, we believe active-with-abort security to be the appropriate security notion in our application to dark pools in traditional financial markets; but not for blockchain based dark pools.

In both protocols addition is a local computation, while multiplication requires communication between the parties. As we are in the pre-processing model, the offline stage is used to produce correlated randomness in order to facilitate the efficient execution of the online phase. This pre-processed data is independent from the input of players, and, to some extent, the function being computed. There are two types of pre-processed data that we will be interested in; multiplication triples and shared random bits.

- Multiplication triples are tuples of the form $(\langle a\rangle, \langle b\rangle, \langle c\rangle)$, such that $c = a \cdot b$. These triples are used in the online phase so as to perform multiplication in only one round of communication. That is, to multiply $\langle x\rangle$ with $\langle y\rangle$, parties calculate locally $\langle \epsilon\rangle \leftarrow \langle x\rangle - \langle a\rangle$ and $\langle \gamma\rangle \leftarrow \langle y\rangle - \langle b\rangle$ then open ϵ and γ. Then the product $\langle x\rangle \cdot \langle y\rangle$ can be obtained by calculating $\langle c\rangle + \epsilon \cdot \langle b\rangle + \gamma \cdot \langle a\rangle + \epsilon \cdot \gamma$ which is a local computation.
- A random bit is a value $\langle r\rangle$ such that r is in $\{0, 1\}$, which is unknown to all the parties. These random bits are used in many protocols, such as the protocol to do comparisons on secret shared values. See for example the protocols given in [Cd10].

As our underlying platform we used the SCALE platform [ACK+21]. This platform supports multi-threading, that is the online phase can be run in multiple threads. Each thread of the online phase is associated with multiple threads generating offline data. Each offline thread is dedicated to generate one type of offline data (triples, bits, etc.). To coordinate these threads a minor extra load is placed on one of the extra MPC parties; which we denote by $\mathcal{P}_0$. This use of threading enables a greater throughput, and to some extent, just in time delivery of the pre-processed data. To understand some of our experimental results, one needs to dig a little deeper into the underlying protocols used and how these are implemented in SCALE. In particular, for some protocol $\mathcal{P}_0$ plays more than a role of coordinating the threads. We now discuss each protocol type in turn.

Shamir Secret Sharing based MPC We assume an honest majority (i.e. $t < l/2$), this has two effects: The underlying secret sharing scheme then acts like an error detecting code, enabling one to detect malicious behaviour. This forms the basis of the protocol in [SW19]: When opening a value, the parties commit first to their views of the shares by adding them to a running hashing value. At the end of the protocol the hash values are compared; if equal then all values have been opened correctly, if different then there has been an error introduced by one of the parties.

The second effect of using honest majority, is that the offline phase (producing the multiplication triples) can be generated by a variant of Maurer's multiplication protocol of [Mau03]. Namely given a random $\langle a \rangle$ and a random $\langle b \rangle$, one can obtain the sharing of $\langle c \rangle$ with $c = a \cdot b$, by resharing the local products. The generation of shared random bits can be performed in a similar manner using standard methodologies.

The question then arises; how to generate a random $\langle a \rangle$. There are two approaches, one which requires interaction and is the naive protocol, and a second one which does not require interaction. The naive protocol we denote by S_2, whilst the one requiring interaction we denote by S_1. The protocol S_1 is based the method Pseudo-Random Secret Sharing (PRSS) from [CDI05]. This PRSS method utilizes pre-shared symmetric keys, which are distributed according to the replicated secret sharing of the underlying access structure. Then to generate a random value each party needs to execute as many PRF calls as one has shares in the underlying replicated secret sharing. Thus the complexity depends on the size of the set Δ in the access structure. For threshold schemes, Δ is of size $\binom{l}{t}$, and thus can become exponentially big. Thus SCALE switches between method S_1 and S_2 as soon as $\binom{l}{t}$ reaches a certain threshold, which is hard wired to $\binom{l}{t} \geq 50$. This helps explain the different behaviour of the algorithms when $\binom{l}{t}$ is small, compared to large, which we will see later.

Full Threshold In this situation we utilize the SPDZ protocol [DPSZ12,DKL+13], which uses Somewhat Homomorphic Encryption in order to produce the data needed in the offline phase. The share values now need a different method of being authenticated, and SPDZ does this using a secret shared MAC key $\alpha \in \mathbb{F}_p$; i.e. each party holds $\alpha_i \in \mathbb{F}_p$ such that $\alpha = \alpha_1 + \ldots + \alpha_l$. A value $x \in \mathbb{F}_p$ is shared both in an additive manner, i.e. $x = x_1 + \ldots + x_l$, but in addition via a sharing of the MAC value $\alpha \cdot x = \gamma_1 + \ldots + \gamma_l$. Thus party $\mathcal{P}_i$ holds both x_i and γ_i. A MAC-Check protocol, see [DKL+13], is used to check that all opened values are consistent with the MACs; and this is done in a manner which reveals neither the MAC values or the MAC key α.

To execute the offline phase, triples (and bits) are generated using Somewhat Homomorphic Encryption. Roughly speaking, while generating a triple, each party $\mathcal{P}_i$ generates random values a_i and b_i, then encrypts using a fully homomorphic scheme and sends the encryption to every other party. Thanks to the homomorphic property, every party can calculate locally $\mathsf{Enc}(c) = (\sum_i \mathsf{Enc}(a_i)) \cdot (\sum_i \mathsf{Enc}(b_i))$. The parties then generate random values f_i and send over the corresponding encryptions, so as to mask $\mathsf{Enc}(c)$ with $\sum_i \mathsf{Enc}(f_i)$. The parties perform then a distributed decryption over $\mathsf{Enc}(c+f)$, and then the parties locally set their shares of c using $c + f$ and f_i. The MAC values on a, b and c are produced using roughly the same manner, using an encryption of $\mathsf{Enc}(\alpha_i)$.

At this point, malicious parties can introduce an additive error to c. To check that this did not happen, the parties "sacrifice" half of the triples, so as to check the correctness of the other half that will be used in the online phase. The sacrifice consists of calculating a random linear combination on the third components of two triples and opening it, then calculating the same combination but now using the first two components of the two triples and opening it. These should be equal if no party cheated. The generation of shared random bits happens in a similar manner.

The distributed decryption of the encrypted values $\mathsf{Enc}(c+f)$ etc is performed in a non-symmetric manner. The partial decryptions are sent to a designated player $\mathcal{P}_0$, who does the combination and then broadcasts the result to the other players. This introduces no security issues, as the correctness is checked via the sacrifice and MAC-Check operations, however, it does mean that this designated player $\mathcal{P}_0$ will receive far more data than it sends. Indeed the amount of data will be a factor l times larger, for the distributed decryption, than the other players. To address this asymmetry, the role of $\mathcal{P}_0$ would rotate between parties after every auction period.

4 Experiments

Our goal is to investigate the performance of a dark market created using an MPC protocol in which the number of participating stake-holders l, i.e. MPC-parties, scales from less than a handful to around one hundred. To do this we simulated a periodic market, where matching occurs at regular intervals with auction time period $t_p = t_o + t_m$, for a universe consisting of a single stock; extensions to more than one stock are immediate and can be accomplished using the methods described in, say, [CST21].

4.1 Methodology

Orders are submitted into the venue during an open period t_o, then matching occurs during period t_m. We vary the number of buy orders N and the number of sell orders M that are submitted during the open period t_o. Buy and sell orders are generated with uniformly random volumes in the range $[1, 100]$. The aim is to understand the effect that liquidity (the number of orders in the system N, M) and number of MPC parties has on time t_p to conduct a single auction. This will allow us to determine the practical limitations on the system.

We implemented the volume match algorithm (see Algorithm 1) using the Scale-Mamba framework between l parties. The MPC volume match algorithm has been analysed previously [CST19, Table 8] for a 2-party and 3-party system, where $l = 2$ and $l = 3$, respectively. It was shown that online and offline runtimes increase with N (the number of sell orders) and M (the number of buy orders), and communication rounds are an increasing function of $min(N, M)$. Here, we simplify the analysis by fixing $N = M$ and comparing two liquidity levels: *low liquidity*, where $N = M = 100$; and *high liquidity*, where $N = M = 500$. Note that we use the terms low and high in a relative sense, as 100 orders per side per auction period for a single stock approximates a very liquid market.

We consider MPC settings with three levels of security threshold:

Shamir min-threshold where $t = 1$, i.e., security is guaranteed only when at most one party is corrupt;

Shamir max-threshold where $t = \lfloor \frac{l-1}{2} \rfloor$, i.e., security is guaranteed unless more than half of all parties are corrupt;

SPDZ full-threshold where $t = l - 1$, i.e., security is guaranteed unless all l parties are corrupt.

Shamir min-threshold and Shamir max-threshold form lower and upper bounds, respectively, for runtimes using the Shamir protocol. Threshold t is a configurable parameter that can take any value within these bounds. In practice, we would not expect a dark pool to use a security threshold with $t = 1$, however we performed this experiment to determine a baseline for the maximum possible speed that the Shamir-based auction protocol can be computed.

We vary l, the number of parties, across the range $l = [2, 100]$ to understand how the dark pool scales and to determine the practical upper bounds of l for each protocol. To simulate a real-world implementation, we used AWS to run the experiments, where each party was assigned to an instance of type C5.4xlarge, containing 16 vCPU and 32 GB in RAM. All l instances were located in the same region (us-west-2). By implementing this cloud-hosted configuration, we are able to better determine real-world runtimes and costs. In practice one could imagine using multiple cloud providers, or have each stakeholder run its own instance locally, so as to avoid potential attacks by a single cloud service provider. This, however, would result in increased latency due to the geographic spread between the different stake-holders. A way-around this problem would be to host the different stake-holder machines in a single data center, with each stakeholder owning and controlling their own machine, but the set of machines hosted in a single location. This is a similar configuration to how high-frequency trading computers are provisioned in some markets.

We measure the online and offline runtimes for a single auction period t_p in markets with high and low liquidity. We also measure communication costs (in MB of data) and the economic costs (i.e. dollar costs) of running a single auction, using current AWS prices. For the online runtimes, each experimental configuration with up to 10 players was run 10 times, and the remaining configurations were run 3 times. Note that variation in runtimes are negligible, so additional repeated trials were not necessary. Offline runtimes were calculated by measuring the time required to generate around four million bits and triples and then scaling the obtained runtimes according to the amount of data needed for each experimental configuration.

We obtained empirical results from a series of experiments that simulate one auction period of the fully distributed MPC dark pool hosted on AWS. We consider online runtimes to conduct the auction (Section 4.2), offline runtimes required to generate necessary data (Section 4.3), the amount of data exchanged between each party during the offline and online phase (Section 4.4), and the per party total economic costs (online and offline) of running one auction on AWS (Section 4.5).

4.2 Online phase - runtimes

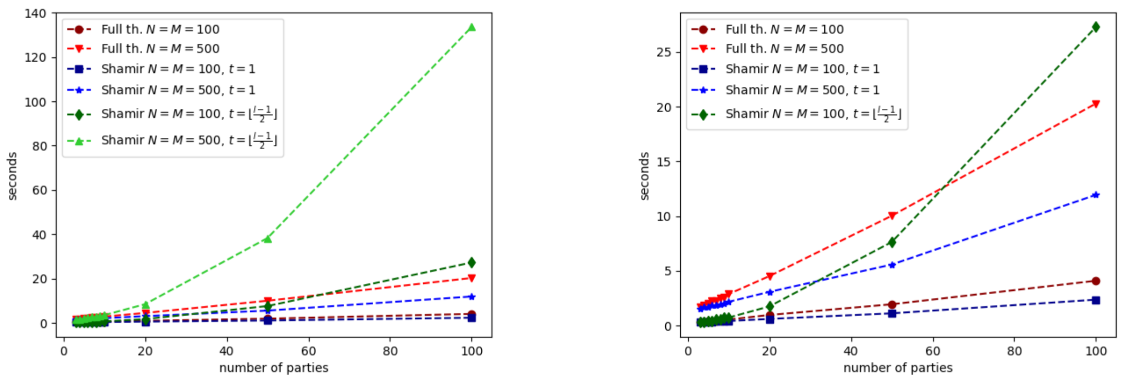

Fig. 3. Online phase runtimes, showing: (left) all protocols; (right) without Shamir $N = M = 500$, $t = \lfloor \frac{l-1}{2} \rfloor$.

Figure 3 presents runtimes corresponding to the online phase of the auction. This represents the minimum time necessary to perform one auction period t_p, which includes the time to accept incoming orders t_o and the time to match t_m. The graph on the left of Figure 3 presents runtimes for all three protocols in both *high liquidity* ($N = M = 500$) and *low liquidity* ($N = M = 100$) markets. We immediately see that as the number of parties is increased, online runtime for Shamir max-threshold increases at a much faster rate than the other protocols. The graph on the right of Figure 3 presents the same information, but with Shamir max-threshold removed for the high liquidity market. Again, we see Shamir max-threshold increase at a faster rate than the other protocols, even in low liquidity markets. With $l = 100$ parties, Shamir max-threshold has runtimes

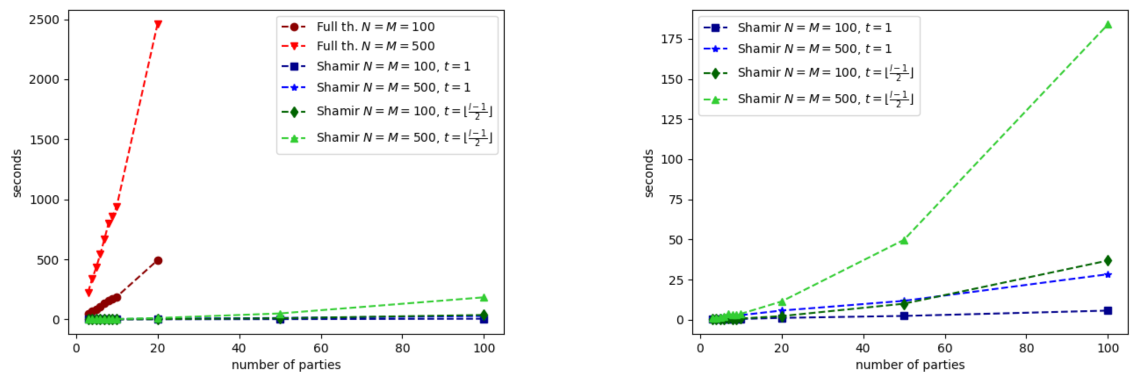

Fig. 4. Offline phase runtimes, showing: (left) all protocols; (right) only Shamir-based protocols.

approximately 11 times greater than Shamir min-threshold. This exponential growth suggests that Shamir secret sharing will not scale to very large numbers of parties, unless the security threshold t is lowered.

In contrast, SPDZ full-threshold displays runtimes that increase almost linearly across the range of parties shown, and in high liquidity markets with $l = 100$ parties SPDZ takes only 20 seconds to perform the online phase, compared with nearly 140 seconds for Shamir max-threshold. This demonstrates that SPDZ full-threshold has a much more efficient online phase than Shamir max-threshold. However, online runtimes for Shamir min-threshold are always lower than runtimes for SPDZ full-threshold. Therefore, by sacrificing some security by lowering t, Shamir secret sharing is able to scale well and can perform more efficiently than SPDZ.

While the experimental configuration is simplistic, these runtimes present promising evidence that the protocols have practical real-world application. For a single stock, SPDZ is able to conduct an auction every 20 seconds, matching $N + M = 1000$ orders with guaranteed security unless all $l = 100$ parties are corrupt. Given that private venues designed for large volume investors tend to have low throughput, often on the order of one trade per stock per day on average (e.g., see statistics presented in [CST21]), this throughput of 3000 trades per minute is more than adequate for a commercial venue.

4.3 Offline phase - runtimes

The offline phase is the time taken to generate the data needed to perform the online phase. This phase can be performed at any time in advance of the online phase, and can be parallelised across multiple machines. Here, we consider the case where there is no parallelisation. This gives us an upper bound on the worst case scenario. Once the offline times become much larger than the online times, then unless parallelisation is used, the protocols become impractical as the necessary data needed to run the online phase cannot be produced in time.

During the offline phase, triples are generated for performing online multiplications and comparisons, and random bits are also generated as they are needed as well for online comparisons. For the case where $N = M = 100$, the offline phase requires the generation of 48,521 triples and 42,105 bits. For the case where $N = M = 500$, the offline phase requires the generation of 242,121 triples

Table 3. Shamir total data sent/received by each player

Protocol	l	N, M	$t = 1$			$t = \lfloor \frac{l-1}{2} \rfloor$		
			Offline		Online	Offline		Online
			Bits	Triples		Bits	Triples	
S_1	3	100	9	9	2	9	9	2
		500	47	43	12	47	43	12
	5	100	16	13	2	19	17	5
		500	78	66	12	94	85	24
S_2	10	100	32	26	2	69	80	9
		500	158	128	12	345	397	47
	20	100	64	51	2	148	170	21
		500	320	255	12	737	849	106
	50	100	313	362	3	383	442	57
		500	1561	1804	13	1910	2204	283
	100	100	630	726	3	775	894	116
		500	3141	3625	15	3868	4462	578

and 210,105 bits. Note that while the absolute number of required triples and bits are fixed for a market configuration, the effort required to generate each triple and bit increases with the number of parties involved.

Figure 4 presents the offline runtimes taken to generate the required number of triples and bits for one auction. The graph on the left shows runtimes for all three protocols in high and low liquidity markets. The graph on the right presents the same information, but with SPDZ full threshold protocol removed. In the left graph, we can clearly see that SPDZ full threshold requires much more time than the Shamir case. This is due to the underlying protocols for SPDZ, where Fully Homomorphic Encryption and zero knowledge proofs are used during the generation of the offline data. For $l = 20$ parties, SPDZ full threshold requires nearly 2500 seconds (approximately 40 minutes) to generate data needed to run one auction in a high liquidity market. In comparison, the online phase for this case takes less than 5 seconds (i.e., 500 times quicker). These runtimes could be aligned by parallelising the offline phase across hundreds of machines, but there are practical issues of scalability to consider here. Also, as the number of parties increases beyond $l > 20$ the experiments for this configuration become unmanageable as the AWS instances we used contained only 32GB RAM. We estimate at least 128GB RAM is required to explore the full range of parties.

In contrast, the graph on the right hand side of Figure 4 shows that the Shamir offline phase is much shorter and scales well, with offline times only slightly larger than online times (e.g., when $N = M = 500$ and $l = 100$, offline time for max-threshold is 180 seconds, online time for the same configuration is around 135 seconds). This affords the opportunity to run the offline phase overnight, before the market opens, or on a parallel machine that generates the triples and bits just-in-time before they are required for the next auction. One should also note that, as described in Section 3.4, for Shamir secret sharing we switch from protocol S_1 to protocol S_2 when $\binom{l}{t} \geq 50$. This can be observed in the graph on the right, where we see that Shamir max-threshold has slower offline phase for case $l = 7$ than for cases $l \in \{8, 9, 10\}$.

Table 4. SPDZ total data sent/received by each player

l	N, M	Offline								Online
		Player $\mathcal{P}_0$				Other Players				
		Bits		Triples		Bits		Triples		
		Sent	Rec.d	Sent	Rec.d	Sent	Rec.d	Sent	Rec.d	
3	100	11	23	14	42	17	11	28	14	3
	500	54	115	70	211	85	54	141	70	16
5	100	22	46	28	85	28	22	42	28	6
	500	108	231	141	422	139	108	211	141	31
10	100	49	105	64	191	55	49	78	64	14
	500	245	522	317	955	276	245	388	317	71
20	100	104	221	135	405	110	104	149	135	30
	500	519	1105	671	2022	550	519	742	671	149

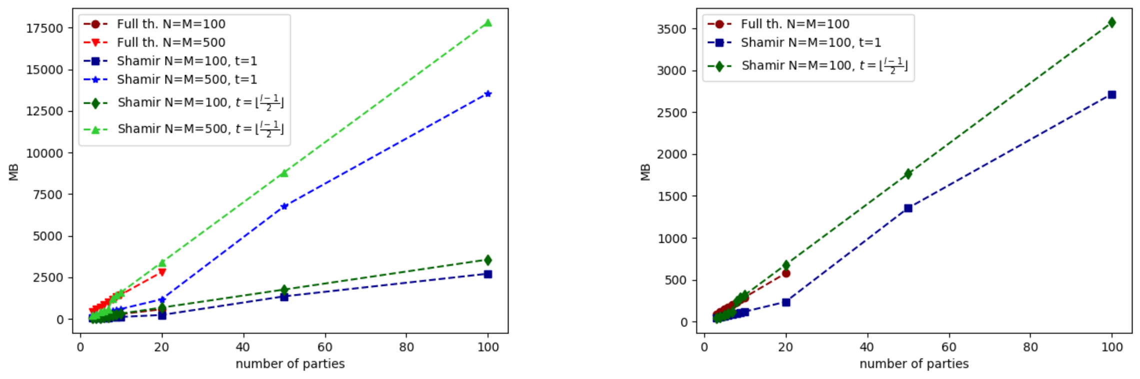

Fig. 5. Total offline data exchanged per party per auction: (left) for all markets; (right) for low liquidity markets only.

4.4 Offline/Online phase - size of data exchanged

Table 3 and Table 4 record the amount of data (nearest MB) each party sends and receives for the triples thread and the bits thread during the offline phase, as well as the data exchanged per party during the online phase, for Shamir-based MPC and for the SPDZ protocol, respectively. The total data exchanged between all parties during the offline phase is presented in Figure 5. Notice that in both protocols, $\mathcal{P}_0$ sends and receives a different amount of data than the other players. In the offline phase, for the Shamir case the difference is negligible. The difference between the amount of data each player sends and receives is also negligible, so in Table 3 we present only one column for data sent/received by $\mathcal{P}_0$/other players. However, for the SPDZ full threshold case, the difference is significant, so all four cases are shown in Table 4. While the data sent by $\mathcal{P}_0$ is slightly less than the data sent by the other players, the data received by $\mathcal{P}_0$ is noticeably greater than the data received by the other players. In a practical dark pool implementation, the party playing the role of $\mathcal{P}_0$ would rotate between parties so that the exchanged data is evenly distributed among the

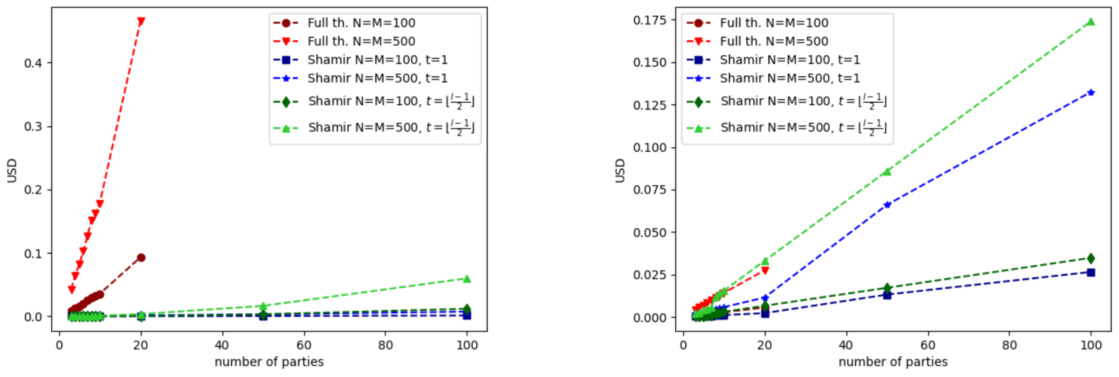

Fig. 6. Total economic cost in USD per party per auction: (left) costs for time; (right) costs for communication.

players. This is the situation considered in Figure 5, where all players in the full threshold protocol exchange the same amount of data.

For the offline phase, in the Shamir case, we distinguish between two cases: (1) when $l < 10$, the S_1 protocol is used; and (2) when $l \geq 10$, the S_2 protocol is used. When S_1 is used for the Shamir case, the overall exchanged data per player (data sent plus received per player) is smaller than the SPDZ full threshold case. When S_2 is used for the Shamir case, the player $\mathcal{P}_0$ still sends more data in the full threshold case than in the Shamir case, however the other players send less data than the Shamir case.

In the online phase, for both the Shamir and the full threshold cases, the difference between data sent/received by $\mathcal{P}_0$/other players is negligible. Thus we represent them in only one column. We also notice that the data exchanged in the online phase, is far less than that exchanged in the offline phase. That is, compared to the offline phase, this data represents between 0.2 and 14 % for the case of Shamir min-threshold, 6 and 14 % for the case of Shamir max-threshold, and 7 and 12 % for the case of SPDZ full-threshold.

4.5 AWS costs per auction

We consider current AWS costs of the region we considered, which are 0.68 USD per instance per hour and 0.01 USD per GB of data sent or received. This communication cost corresponds to the case where players hold their instances in different regions or different availability zones since communication within the same availability zone is not charged. If there is a large number of players, we will not have as many availability zones as players, and therefore the values presented here represent an upper bound for the cost of communication.

The time cost and the communication cost per party for one auction are represented in Figure 6. We notice that in the Shamir case, the communication cost is higher than the time cost in the offline phase, while the time cost is higher than the communication cost in online phase, wheras in the full threshold case, the time cost is higher than the communication cost in the offline phase, while the communication cost is higher than the time cost in online phase. The key take-away is that the cost per auction, per stock, is under one dollar when using Shamir based protocols. Given the size of the trades on commercial dark-market platforms, where each trade can be many millions of dollars, the cost of running the market in the dark becomes negligible in this case even for 100 parties. If one restricted to 20 stakeholders, then the cost for using full threshold protocols is also sub one dollar, and thus acceptable.

5 Discussion

In the popular consciousness, financial trading venues invoke images of vast quantities of fleeting order flow driven by high frequency trading (HFT) algorithms. However, while this is true for many venues (e.g., see [DC18]), it is not the case for all. For example, LiquidMetrix reported that Liquidnet executed 7298 trades across more than 1000 instruments during February 2017; i.e., fewer than one trade per instrument per trading day. We have shown that $l = 100$ participating parties can handle 1000 orders in 140 seconds (using the slowest configuration of Shamir honesty majority) at a cost to each party of around 24 cents; and fewer orders reduces computation time and economic costs. These results suggest that the "no operator" Liquidnet-style architecture is economically and computationally viable. If the dark pool runs a periodic volume match every, say, 5 minutes, the platform will be able to easily handle expected order flow and dollar costs per participant will be a vanishing fraction of the total value traded. To enable trading across many stocks, the secure *gateway allocation* method introduced in [CST21, Section 4.1] can be used.

If the MPC dark pool architecture that we have studied in this paper were to be instantiated on a public blockchain, there remain potential issues of trader anonymity and the use of security-with-abort (see Table 2). We have not attempted to address these challenges as traditional financial trading venues are not implemented on a blockchain. However, they offer interesting avenues of investigation for future work.

Acknowledgements

This work has been supported in part by the FWO under an Odysseus project GOH9718N, and by CyberSecurity Research Flanders with reference number VR20192203.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of any of the funders.

References

ABPV20. Gilad Asharov, Tucker Balch, Antigoni Polychroniadou, and Manuela Veloso. Privacy-preserving dark pools. In *AAMAS '20: 19th International Conference on Autonomous Agents and Multiagent Systems*, pages 1747–1749, Auckland, New Zealand, 2020. International Foundation for Autonomous Agents and Multiagent Systems.

ABPV21. Gilad Asharov, Tucker Richard Balch, Antigoni Polychroniadou, and Manuela Veloso. Methods and systems for implementing privacy-preserving dark pools. World Intellectual Property Organization Patent WO/2021/050088, filed Sep 16, 2019, published Mar 18, 2021, mar 2021.

ACK$^+$21. Abdelrahaman Aly, Kelong Cong, Marcel Keller, Emmanuela Orsini, Dragos Rotaru, Oliver Scherer, Peter Scholl, Nigel P. Smart, Titouan Tanguy, and Tim Wood. SCALE and MAMBA v1.14: Documentation, 2021.

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, Michael I. Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In Roger Dingledine and Philippe Golle, editors, *FC 2009: 13th International Conference on Financial Cryptography and Data Security*, volume 5628 of *Lecture Notes in Computer Science*, pages 325–343, Accra Beach, Barbados, February 23–26, 2009. Springer, Heidelberg, Germany.

BDF21. Carsten Baum, Bernardo David, and Tore Kasper Frederiksen. P2DEX: Privacy-preserving decentralized cryptocurrency exchange. In Kazue Sako and Nils Ole Tippenhauer, editors, *ACNS 21: 19th International Conference on Applied Cryptography and Network Security, Part I*, volume 12726 of *Lecture Notes in Computer Science*, pages 163–194, Kamakura, Japan, June 21–24, 2021. Springer, Heidelberg, Germany.

BDJ$^+$06. Peter Bogetoft, Ivan Damgård, Thomas Jakobsen, Kurt Nielsen, Jakob Pagter, and Tomas Toft. A practical implementation of secure auctions based on multiparty integer computation. In Giovanni Di Crescenzo and Avi Rubin, editors, *FC 2006: 10th International Conference on Financial Cryptography and Data Security*, volume 4107 of *Lecture Notes in Computer Science*, pages 142–147, Anguilla, British West Indies, February 27 – March 2, 2006. Springer, Heidelberg, Germany.

BDP20. Tucker Balch, Benjamin E. Diamond, and Antigoni Polychroniadou. Secretmatch: inventory matching from fully homomorphic encryption. In *ICAIF '20: The First ACM International Conference on AI in Finance*, pages 15:1–15:7, New York, NY, USA, 2020. ACM.

BHSR20. Samiran Bag, Feng Hao, Siamak F. Shahandashti, and Indranil Ghosh Ray. SEAL: Sealed-bid auction without auctioneers. *IEEE Transactions on Information Forensics and Security*, 15:2042–2052, 2020.

ByCD$^+$21. Carsten Baum, James Hsin yu Chiang, Bernardo David, Tore Kasper Frederiksen, and Lorenzo Gentile. SoK: Mitigation of front-running in decentralized finance. Cryptology ePrint Archive, Report 2021/1628, 2021. https://eprint.iacr.org/2021/1628.

Cd10. Octavian Catrina and Sebastiaan de Hoogh. Improved primitives for secure multiparty integer computation. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10: 7th International Conference on Security in Communication Networks*, volume 6280 of *Lecture Notes in Computer Science*, pages 182–199, Amalfi, Italy, September 13–15, 2010. Springer, Heidelberg, Germany.

CDI05. Ronald Cramer, Ivan Damgård, and Yuval Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 342–362, Cambridge, MA, USA, February 10–12, 2005. Springer, Heidelberg, Germany.

CGH$^+$18. Koji Chida, Daniel Genkin, Koki Hamada, Dai Ikarashi, Ryo Kikuchi, Yehuda Lindell, and Ariel Nof. Fast large-scale honest-majority MPC for malicious adversaries. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 34–64, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

CST19. John Cartlidge, Nigel P. Smart, and Younes Talibi Alaoui. MPC joins the dark side. In Steven D. Galbraith, Giovanni Russello, Willy Susilo, Dieter Gollmann, Engin Kirda, and Zhenkai Liang, editors, *ASIACCS 19: 14th ACM Symposium on Information, Computer and Communications Security*, pages 148–159, Auckland, New Zealand, July 9–12, 2019. ACM Press.

CST21. John Cartlidge, Nigel P. Smart, and Younes Talibi Alaoui. Multi-party computation mechanism for anonymous equity block trading: A secure implementation of turquoise plato uncross. *Intelligent Systems in Accounting, Finance and Management*, 28(4):239–267, 2021.

DC18. Matthew Duffin and John Cartlidge. Agent-based model exploration of latency arbitrage in fragmented financial markets. In *IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2312–2320, 2018.

dGCP$^+$21. Mariana Botelho da Gama, John Cartlidge, Antigoni Polychroniadou, Nigel P. Smart, and Younes Talibi Alaoui. Kicking-the-bucket: Fast privacy-preserving trading using buckets. Cryptology ePrint Archive, Report 2021/1549, 2021. https://eprint.iacr.org/2021/1549. To appear in *FC'22: Financial Cryptography and Data Security*. https://fc22.ifca.ai/preproceedings/27.pdf.

DKL$^+$13. Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *ESORICS 2013: 18th European Symposium on Research in Computer Security*, volume 8134 of *Lecture Notes in Computer Science*, pages 1–18, Egham, UK, September 9–13, 2013. Springer, Heidelberg, Germany.

DPSZ12. Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.

GVJR21. Kavya Govindarajan, Dhinakaran Vinayagamurthy, Praveen Jayachandran, and Chester Rebeiro. Privacy-preserving decentralized exchange marketplaces. ePrint arXiv:2111.15259v2. Dec 2021, December 2021.

HW22. Lioba Heimbach and Roger Wattenhofer. Sok: Preventing transaction reordering manipulations in decentralized finance. ePrint arXiv:2203.11520. Mar 2022, March 2022.

JSL21. Robin Jadoul, Nigel P. Smart, and Barry Van Leeuwen. MPC for Q_2 access structures over rings and fields. In Riham AlTawy and Andreas Hülsing, editors, *Selected Areas in Cryptography - 28th International Conference, SAC 2021, Virtual Event, September 29 - October 1, 2021, Revised Selected Papers*, volume 13203 of *Lecture Notes in Computer Science*, pages 131–151. Springer, 2021.

KRSW18. Marcel Keller, Dragos Rotaru, Nigel P. Smart, and Tim Wood. Reducing communication channels in MPC. In Dario Catalano and Roberto De Prisco, editors, *SCN 18: 11th International Conference on Security in Communication Networks*, volume 11035 of *Lecture Notes in Computer Science*, pages 181–199, Amalfi, Italy, September 5–7, 2018. Springer, Heidelberg, Germany.

Mau03. Ueli M. Maurer. Secure multi-party computation made simple (invited talk). In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02: 3rd International Conference on Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 14–28, Amalfi, Italy, September 12–13, 2003. Springer, Heidelberg, Germany.

MN21. Fabio Massacci and Chan Nam Ngo. Distributed financial exchanges: Security challenges and design principles. *IEEE Security & Privacy*, 19(1):54–64, 2021.

MNN$^+$17. Fabio Massacci, Chan Nam Ngo, Jing Nie, Daniele Venturi, and Julian Williams. The seconomics (security-economics) vulnerabilities of decentralized autonomous organizations. In Frank Stajano, Jonathan Anderson, Bruce Christianson, and Vashek Matyáš, editors, *Security Protocols XXV*, volume 10476 of *LNCS*, pages 171–179, Cham, May 2017. Springer.

MNN$^+$18. Fabio Massacci, Chan Nam Ngo, Jing Nie, Daniele Venturi, and Julian Williams. FuturesMEX: Secure, distributed futures market exchange. In *2018 IEEE Symposium on Security and Privacy*, pages 335–353, San Francisco, CA, USA, May 21–23, 2018. IEEE Computer Society Press.

Ngo19. Chan Nam Ngo. *Secure, Distributed Financial Exchanges: Design and Implementation.* PhD thesis, Department of Information Engineering and Computer Science, University of Trento, Italy, oct 2019.

NMKW21. Nam Ngo, Fabio Massacci, Florian Kerschbaum, and Julian Williams. Practical witness-key-agreement for blockchain-based dark pools financial trading. In N. Borisov and C. Diaz, editors, *FC'21: Financial Cryptography and Data Security*, volume 12675 of *LNCS*, pages 579–598, Cham, 2021. Springer.

SW19. Nigel P. Smart and Tim Wood. Error detection in monotone span programs with application to communication-efficient multi-party computation. In Mitsuru Matsui, editor, *Topics in Cryptology – CT-RSA 2019*, volume 11405 of *Lecture Notes in Computer Science*, pages 210–229, San Francisco, CA, USA, March 4–8, 2019. Springer, Heidelberg, Germany.

Uni22. United States of America before the Securities and Exchange Commission. In the Matter of tZERO ATS LLC, Exchange Act of 1934, Release No. 93938, 10 Jan 2022. https://www.sec.gov/litigation/admin/2022/34-93938.pdf, January 2022.

A Experimental results

In this Appendix we give the precise numerical values used to produce the graphs in the main body.

Table 5. Online phase in seconds with Shamir based MPC

l	N	M	Online time $t = 1$	Online time $t = \lfloor \frac{l-1}{2} \rfloor$
3	100	100	0.29	0.29
	500	500	1.52	1.52
4	100	100	0.30	0.30
	500	500	1.65	1.65
5	100	100	0.33	0.39
	500	500	1.65	2.14
6	100	100	0.34	0.43
	500	500	1.81	2.26
7	100	100	0.37	0.53
	500	500	1.81	2.64
8	100	100	0.37	0.57
	500	500	1.89	2.75
9	100	100	0.41	0.71
	500	500	2.00	3.22
10	100	100	0.41	0.72
	500	500	2.16	3.51
20	100	100	0.60	1.74
	500	500	3.07	8.46
50	100	100	1.11	7.63
	500	500	5.57	38.32
100	100	100	2.36	27.29
	500	500	11.94	133.6

Table 6. Online phase in seconds with full threshold

l	N	M	Online time
3	100	100	0.3
	500	500	1.68
4	100	100	0.33
	500	500	1.81
5	100	100	0.37
	500	500	2.02
6	100	100	0.42
	500	500	2.23
7	100	100	0.43
	500	500	2.27
8	100	100	0.49
	500	500	2.46
9	100	100	0.52
	500	500	2.55
10	100	100	0.54
	500	500	2.86
20	100	100	0.96
	500	500	4.51
50	100	100	1.94
	500	500	10.03
100	100	100	4.09
	500	500	20.29

Table 7. Offline phase in seconds with Shamir based MPC

l	N	M	Offline time $t = 1$	Offline time $t = \lfloor \frac{l-1}{2} \rfloor$
3	100	100	0.21	0.21
	500	500	1.06	1.06
4	100	100	0.24	0.24
	500	500	1.18	1.18
5	100	100	0.26	0.33
	500	500	1.31	1.66
6	100	100	0.30	0.45
	500	500	1.49	2.26
7	100	100	0.35	0.77
	500	500	1.72	3.73
8	100	100	0.42	0.63
	500	500	2.08	3.14
9	100	100	0.46	0.68
	500	500	2.30	3.39
10	100	100	0.47	0.73
	500	500	2.83	3.63
20	100	100	1.14	2.27
	500	500	5.70	11.35
50	100	100	2.36	9.97
	500	500	11.76	49.73
100	100	100	5.68	36.87
	500	500	28.33	183.99

Table 8. Offline phase in seconds with full threshold. Note for more than 50 parties the offline time became too-prohibitive to compute.

l	N	M	Offline time
3	100	100	45
	500	500	224
4	100	100	68
	500	500	337
5	100	100	87
	500	500	433
6	100	100	109
	500	500	543
7	100	100	134
	500	500	667
8	100	100	160
	500	500	798
9	100	100	172
	500	500	859
10	100	100	188
	500	500	938
20	100	100	492
	500	500	2457
50	100	100	-
	500	500	-
100	100	100	-
	500	500	-

Table 9. Offline and online Data Sent/Received in MB by each player with Shamir based MPC

l	N	M	$t = 1$			$t = \lfloor \frac{l-1}{2} \rfloor$		
			Offline		Online	Offline		Online
			Bits	Triples		Bits	Triples	
3	100	100	9.46	8.53	2.36	9.46	8.53	2.36
	500	500	47.19	42.56	11.79	47.19	42.56	11.79
4	100	100	12.75	10.9	2.37	12.75	10.9	2.37
	500	500	63.61	54.38	11.82	63.61	54.38	11.82
5	100	100	15.62	13.27	2.38	18.91	17.06	4.73
	500	500	77.97	66.20	11.85	94.38	85.12	23.59
6	100	100	18.91	15.64	2.38	22.2	19.43	4.73
	500	500	94.38	78.03	11.88	110.8	96.94	23.62
7	100	100	22.20	18.48	2.39	28.37	25.59	7.09
	500	500	110.80	92.21	11.91	141.57	127.68	35.38
8	100	100	25.49	20.85	2.40	53.45	61.6	7.10
	500	500	127.21	104.04	11.94	266.73	307.38	35.41
9	100	100	28.37	23.22	2.40	62.91	72.5	9.46
	500	500	141.57	115.86	11.97	313.93	361.76	47.18
10	100	100	31.66	25.59	2.41	69.08	79.6	9.46
	500	500	157.99	127.68	12.00	344.7	397.23	47.21
20	100	100	64.14	51.17	2.47	147.61	170.11	21.28
	500	500	320.08	255.36	12.30	736.6	848.84	106.18
50	100	100	312.91	361.54	2.66	382.81	441.62	56.74
	500	500	1561.42	1804.09	13.21	1910.23	2203.68	283.09
100	100	100	629.52	726.39	2.97	775.08	894.13	115.84
	500	500	3141.32	3624.72	14.71	3867.66	4461.74	577.93

Table 10. Offline and online Data Sent and Received in MB by each player for full Threshold

l	N	M	Offline								Online
			$\mathcal{P}_0$				Other Players				
			Bits		Triples		Bits		Triples		
			Sent	Received	Sent	Received	Sent	Received	Sent	Received	
3	100	100	10.87	23.12	14.07	42.32	17.0	10.87	28.19	14.07	3.15
	500	500	54.24	115.35	70.19	211.18	84.83	54.24	140.69	70.19	15.73
4	100	100	16.31	34.68	21.12	63.46	22.42	16.31	35.24	21.12	4.73
	500	500	81.37	173.06	105.37	316.69	111.89	81.37	175.86	105.37	23.59
5	100	100	21.74	46.23	28.16	84.64	27.86	21.73	42.26	28.16	6.31
	500	500	108.49	230.7	140.54	422.35	139.01	108.43	210.88	140.54	31.45
6	100	100	27.28	58.07	35.3	106.29	33.43	27.28	49.52	35.3	7.88
	500	500	136.12	289.75	176.15	530.38	166.84	136.12	247.09	176.15	39.32
7	100	100	32.73	69.67	42.38	127.52	38.88	32.73	56.56	42.38	9.46
	500	500	163.31	347.65	211.47	636.34	194.02	163.31	282.26	211.47	47.18
8	100	100	38.19	81.29	49.46	148.79	44.34	38.19	63.64	49.43	11.04
	500	500	190.56	405.62	246.79	742.44	221.27	190.56	317.58	246.64	55.05
9	100	100	43.64	92.9	56.51	170.05	49.79	43.64	70.69	56.51	12.61
	500	500	217.75	463.58	281.96	848.55	248.46	217.75	352.75	281.96	62.91
10	100	100	49.1	104.52	63.58	191.28	55.25	49.1	77.77	63.55	14.19
	500	500	245.0	521.54	317.28	954.5	275.71	245.0	388.07	317.13	70.77
20	100	100	103.94	221.38	134.54	405.22	110.12	103.94	148.79	134.54	29.95
	500	500	518.66	1104.71	671.36	2022.06	549.5	518.66	742.44	671.36	149.41
50	100	100	-	-	-	-	-	-	-	-	77.25
	500	500	-	-	-	-	-	-	-	-	385.32
100	100	100	-	-	-	-	-	-	-	-	156.07
	500	500	-	-	-	-	-	-	-	-	778.51

Table 11. Cost in USD for Time and Communication for one player to run the offline phase in the Shamir setting.

l	N	M	$t = 1$		$t = \lfloor \frac{l-1}{2} \rfloor$	
			Time	Communication	Time	Communication
3	100	100	3.97e-05	3.51e-04	3.97e-05	3.51e-04
	500	500	2.00e-04	1.75e-03	2.00e-04	1.75e-03
4	100	100	4.53e-05	4.62e-04	4.53e-05	4.62e-04
	500	500	2.23e-04	2.30e-03	2.23e-04	2.30e-03
5	100	100	4.91e-05	5.64e-04	6.23e-05	7.03e-04
	500	500	2.47e-04	2.82e-03	3.14e-04	3.51e-03
6	100	100	5.67e-05	6.75e-04	8.50e-05	8.13e-04
	500	500	2.81e-04	3.37e-03	4.27e-04	4.06e-03
7	100	100	6.61e-05	7.95e-04	1.45e-04	1.05e-03
	500	500	3.25e-04	3.97e-03	7.05e-04	5.26e-03
8	100	100	7.93e-05	9.05e-04	1.19e-04	2.25e-03
	500	500	3.93e-04	4.52e-03	5.93e-04	1.12e-02
9	100	100	8.69e-05	1.01e-03	1.28e-04	2.64e-03
	500	500	4.34e-04	5.03e-03	6.40e-04	1.32e-02
10	100	100	8.88e-05	1.12e-03	1.38e-04	2.90e-03
	500	500	5.35e-04	5.58e-03	6.86e-04	1.45e-02
20	100	100	2.15e-04	2.25e-03	4.29e-04	6.21e-03
	500	500	1.08e-03	1.12e-02	2.13e-03	3.10e-02
50	100	100	4.46e-04	1.32e-02	1.88e-03	1.61e-02
	500	500	2.22e-03	6.57e-02	9.39e-03	8.03e-02
100	100	100	1.07e-03	2.65e-02	6.96e-03	3.26e-02
	500	500	5.35e-03	1.32e-01	3.48e-02	1.63e-01

Table 12. Cost in USD for Time and Communication for one player to run the online phase in the Shamir setting.

l	N	M	$t = 1$		$t = \lfloor \frac{l-1}{2} \rfloor$	
			Time	Communication	Time	Communication
3	100	100	5.48e-05	4.61e-05	5.48e-05	4.61e-05
	500	500	2.87e-04	2.30e-04	2.87e-04	2.30e-04
4	100	100	5.67e-05	4.63e-05	5.67e-05	4.63e-05
	500	500	3.12e-04	2.31e-04	3.12e-04	2.31e-04
5	100	100	6.23e-05	4.65e-05	7.37e-05	9.24e-05
	500	500	3.12e-04	2.31e-04	4.04e-04	4.61e-04
6	100	100	6.42e-05	4.65e-05	8.12e-05	9.24e-05
	500	500	3.42e-04	2.31e-04	4.27e-04	4.61e-04
7	100	100	6.99e-05	4.67e-05	1.00e-04	1.38e-04
	500	500	3.42e-04	2.33e-04	4.99e-04	6.91e-04
8	100	100	6.99e-05	4.69e-05	1.08e-04	1.39e-04
	500	500	3.57e-04	2.33e-04	5.19e-04	6.92e-04
9	100	100	7.74e-05	4.69e-05	1.34e-04	1.85e-04
	500	500	3.78e-04	2.34e-04	6.08e-04	9.21e-04
10	100	100	7.74e-05	4.71e-05	1.36e-04	1.85e-04
	500	500	4.08e-04	2.34e-04	6.63e-04	9.22e-04
20	100	100	1.13e-04	4.82e-05	3.29e-04	4.16e-04
	500	500	5.80e-04	2.40e-04	1.60e-03	2.07e-03
50	100	100	2.10e-04	5.20e-05	1.44e-03	1.11e-03
	500	500	1.05e-03	2.58e-04	7.24e-03	5.53e-03
100	100	100	4.46e-04	5.80e-05	5.15e-03	2.26e-03
	500	500	2.26e-03	2.87e-04	2.52e-02	1.13e-02

Table 13. Cost in USD for Time and Communication for one player to run the offline phase in the full threshold setting.

l	N	M	Time	Communication
3	100	100	8.50e-03	7.51e-04
	500	500	4.23e-02	3.75e-03
4	100	100	1.28e-02	1.03e-03
	500	500	6.37e-02	5.13e-03
5	100	100	1.64e-02	1.29e-03
	500	500	8.18e-02	6.44e-03
6	100	100	2.06e-02	1.55e-03
	500	500	1.03e-01	7.75e-03
7	100	100	2.53e-02	1.81e-03
	500	500	1.26e-01	9.02e-03
8	100	100	3.02e-02	2.06e-03
	500	500	1.51e-01	1.03e-02
9	100	100	3.25e-02	2.31e-03
	500	500	1.62e-01	1.15e-02
10	100	100	3.55e-02	2.56e-03
	500	500	1.77e-01	1.28e-02
20	100	100	9.29e-02	5.04e-03
	500	500	4.64e-01	2.45e-02

Table 14. Cost in USD for Time and Communication for one player to run the online phase in the full threshold setting

l	N	M	Time	Communication
3	100	100	5.67e-05	6.15e-05
	500	500	3.17e-04	3.07e-04
4	100	100	6.23e-05	9.24e-05
	500	500	3.42e-04	4.61e-04
5	100	100	6.99e-05	1.23e-04
	500	500	3.82e-04	6.14e-04
6	100	100	7.93e-05	1.54e-04
	500	500	4.21e-04	7.68e-04
7	100	100	8.12e-05	1.85e-04
	500	500	4.29e-04	9.21e-04
8	100	100	9.26e-05	2.16e-04
	500	500	4.65e-04	1.08e-03
9	100	100	9.82e-05	2.46e-04
	500	500	4.82e-04	1.23e-03
10	100	100	1.02e-04	2.77e-04
	500	500	5.40e-04	1.38e-03
20	100	100	1.81e-04	5.85e-04
	500	500	8.52e-04	2.92e-03
50	100	100	3.66e-04	1.51e-03
	500	500	1.89e-03	7.53e-03
100	100	100	7.73e-04	3.05e-03
	500	500	3.83e-03	1.52e-02