Correlated-Output-Differential-Privacy and Applications to Dark Pools

James Hsin-yu Chiang
Technical University of Denmark, Denmark

Bernardo David
IT University of Copenhagen, Denmark

Mariana Gama
imec-COSIC, KU Leuven, Belgium

Christian Janos Lebeda
IT University of Copenhagen, Denmark
Basic Algorithms Research Copenhagen, Denmark

Abstract

In the classical setting of differential privacy, a privacy-preserving query is performed on a private database, after which the query result is released to the analyst; a differentially private query ensures that the presence of a single database entry is protected from the analyst's view. In this work, we contribute the first definitional framework for differential privacy in the *trusted curator setting* (Fig. 1); clients submit private inputs to the trusted curator which then computes individual outputs privately returned to each client. The adversary is more powerful than the standard setting; it can corrupt up to $n-1$ clients and subsequently decide inputs and learn outputs of corrupted parties. In this setting, the adversary also obtains leakage from the honest output that is *correlated with a corrupted output*. Standard differentially private mechanisms protect client inputs but do not mitigate output correlation leaking arbitrary client information, which can forfeit client privacy completely. We initiate the investigation of a novel notion of *correlated-output-differential-privacy* to bound the leakage from output correlation in the trusted curator setting. We define the satisfaction of both standard and correlated-output-differential-privacy as *round-differential-privacy* and highlight the relevance of this novel privacy notion to all application domains in the trusted curator model.

We explore *round-differential-privacy* in traditional "dark pool" market venues, which promise privacy-preserving trade execution to mitigate front-running; privately submitted trade orders and trade execution are kept private by the trusted venue operator. We observe that dark pools satisfy neither classic *nor* correlated-output-differential privacy; in markets with low trade activity, the adversary may trivially observe recurring, honest trading patterns, and anticipate and front-run future trades. In response, we present the first *round-differentially-private* market mechanisms that formally mitigate information leakage from all trading activity of a user. This is achieved with *fuzzy order matching*, inspired by the standard randomized response mechanism; however, this also introduces a liquidity mismatch as buy and sell orders are not guaranteed to execute pairwise, thereby weakening output correlation; this mismatch is compensated for by a round-differentially-private *liquidity provider* mechanism, which freezes a noisy amount of assets from the liquidity provider for the duration of a privacy epoch, but leaves trader balances unaffected. We propose oblivious algorithms for realizing our proposed market mechanisms with secure multi-party computation (MPC) and implement these in the Scale-Mamba Framework using Shamir Secret Sharing based MPC. We demonstrate practical, round-differentially-private trading with comparable throughput as prior work implementing (traditional) dark pool algorithms in MPC; our experiments demonstrate practicality for both traditional finance and decentralized finance settings.

2012 ACM Subject Classification Security and privacy → Privacy-preserving protocols

Keywords and phrases Differential Privacy, Secure Multi-party Computation, Dark Pools, Decentralized Finance

Digital Object Identifier 10.4230/LIPIcs...

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

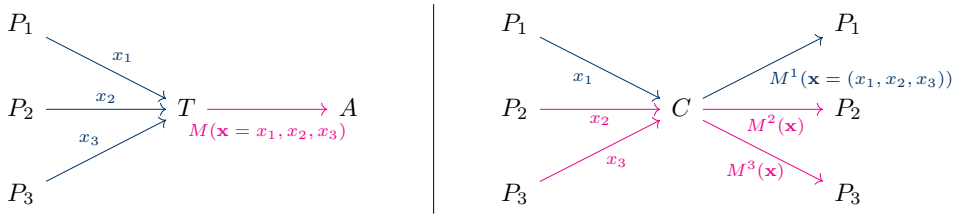

Figure 1 The standard model of differential privacy (L) vs. the trusted curator model (R). In this work, we contribute the first definitional framework for differential privacy in the trusted curator model (§3.1).

1 Introduction

In the standard differential privacy setting (Fig. 1, left), a single analyst (A) receives a query on private inputs from clients (P_1, P_2, P_3) computed by the trusted third party (T). A differentially private query protects the privacy of an input x_i submitted by client P_i. In the trusted curator model (Fig. 1, right), the curator C evaluates a function on all privately submitted inputs, $(y_1, y_2, y_3) \leftarrow M(x_1, x_2, x_3)$, and returns each output y_i privately to client P_i, which may be corrupted by the adversary. A (classically) differentially private mechanism M will protect the honest input x_1. However, if honest output $y_1 = M^1(x_1, x_2, x_3)$ and adversarial output $y_{i\neq 1} = M^i(x_1, x_2, x_3)$ are correlated, honest y_1 may be trivially inferred from an adversarial $y_{i\neq 1}$, breaking client privacy. In this work, we introduce *correlated-output differential privacy* (§3.3) to protect against such leakage to achieve client privacy in the setting of the trusted curator. The conjunction with standard differential privacy protecting inputs is defined as *round-differential-privacy* (§3.4), protecting the entire client transcript in each interaction round. In this model, the adversary can inject inputs to each round; *round-differential-privacy* insures that such a chosen-input attack has a bounded effect on the honest user's output. We highlight the investigation of round-differentially-private algorithms for general and specific application domains as a research question of independent interest. In this work, we investigate round-differentially-private market applications to prevent front-running in traditional or decentralized finance.

The term front-running originates from the notion of "getting in front" of pending trades. A party anticipating a large buy order may purchase the same asset first, as the pending large buy order will likely drive up the price of the asset; the front-running party can then sell the asset at a higher price following the execution of the large buy order. Front-running occurs whenever submitted trade orders that have yet to be executed are observable by the front-running adversary. In traditional finance, the presence of pending orders may be public or inferred from market order books. In decentralized finance, pending transactions are publicly gossiped across a peer-to-peer network. In both settings, front-running is prevalent.

In traditional finance, Dark Pool venues [24] promise the private execution of trades. Here, clients submit private orders to the venue operator, who then computes the execution of trades without leaking pending orders submitted by clients; pre-trade privacy ensures that pending orders remain private, whilst post-trade privacy protects the privacy of the trade execution. An initial attempt to provide both pre-trade and post-trade privacy would be to implement a market venue with a trusted curator role; in each round, parties can submit the trade orders privately to the curator, which then computes an optimal matching of submitted trade orders. The trade results are then privately output to each participating client. Since all communication between individual client and curator is private, no trade orders or trade

outcomes are *explicitly* leaked. Removing the trusted curator role in dark pools with secure multi-party computation (MPC) has been recently explored in [6, 7, 11, 12], demonstrating practical run-times amenable to real-world trading volume.

However, we observe that *round-differential-privacy* is violated in classical order-matching algorithms run by dark pool venues; a trade execution always implies a counter-party, thus revealing the presence of another trade in the opposing direction (Lemmas 4 and 6); such leakage occurs because trade execution are necessarily *correlated* between different parties. In venues with low trade volume, such inferences may lead to practical attacks. Consider the Time-Weighted Average Price (TWAP)[1] trade, where a larger trade volume is scheduled as smaller trades over time to minimize price impact. If the periodic execution of such smaller trades is detected early, the remaining trade schedule can be anticipated and front-run. This motivates our investigation of market mechanisms with formal, privacy guarantees protecting the full transcript of interactions between trader and dark pool venue.

In this work, we contribute round-differentially-private (§4) market mechanisms for the trusted curator model. Here, the actual "trade", "no-trade" outcome for each order is determined by *sampling* a Bernoulli distribution biased towards the deterministically computed, optimal matching. However, since trades are filled or not filled based on independently sampling trade outcomes, there is no guarantee that each executed trade is matched with an equivalent volume in opposing direction; therefore, a liquidity deficit may occur. Here, market makers or *liquidity providers* make up for liquidity deficits. To prevent liquidity providers from learning about the traded volume of a single user from their updated liquidity balances, a random, yet bounded amount of market maker liquidity is *frozen* to obtain round differential privacy; frozen liquidity is returned to the market makers at the end of each privacy epoch, that is sufficiently long for honest users to complete long-running trade strategies.

We instantiate our fuzzy order matching market mechanisms with oblivious MPC algorithms (§5) and implement these algorithms using the Scale-Mamba framework [3] with Shamir Secret Sharing based MPC; history has shown that dark pool venue operators frequently exploit confidential order flow information [19, 20, 18], thus motivating us to demonstrate practical feasibility of distributing round-differential private market mechanisms across MPC committees in lieu of a trusted operator. We show that our algorithms not only satisfy stronger privacy guarantees than considered in previous work, they also achieve high order throughput appropriate for most real-world settings. In fact, even with the additional overhead induced by round-differentially-private algorithms, we obtain runtimes comparable to the algorithms presented in prior work implementing traditional dark pools in MPC [11]. Finally, we emphasize that our fair market mechanism designs are applicable to both traditional dark pool venue operators and decentralized finance. Our fair markets can be instantiated in privacy-preserving smart contract frameworks realized by a MPC committee and privacy-preserving ledger, most recently demonstrated by Baum et al. in [4] with minimal complexity overhead; here, trade execution is settled in private on a public ledger.

1.1 Related Work

Differentially private markets. Chitra *et al.* [10] propose a *Uniform Random Execution* algorithm which permutes and splits submitted trades in a randomized, differentially private

[1] `https://en.wikipedia.org/wiki/Time-weighted_average_price`

manner. We note that [10] does not offer output or *post-trade* privacy; all executed trades are seen by the adversary. Thus, this approach does not contribute to our goal of protecting the privacy of long-running trader strategies performed over multiple rounds.

Dark pool markets. Recent proposals [6, 7, 11, 12] have convincingly demonstrated that the role distribution of the dark pool operator can be instantiated in practice with multi-party computation (MPC) to prevent abuse of private order information. Still, these works also do not consider the entirety of information flow leaking from all honest trader activity; Firstly, adversarial outputs reveal information about privately submitted honest inputs (Lemma 4) and secondly, outputs are correlated, such that an adversary also obtains information about honest outputs (Lemma 6). In the decentralized finance setting, homomorphic encryption has been proposed to aggregate orders obliviously [23]; however, since all inputs are encrypted to the same public key, any subsequent decryption to reveal the aggregated order will leak privacy of any single trade, if all but one client has been corrupted.

Differential privacy and MPC. Whilst differentially private mechanisms have been implemented in MPC, these works do not consider privacy over the full, individual transcript in the trusted curator model (§3.1), where clients submit private inputs and receive private outputs. Instead, the MPC output is a single query result computed over inputs from a private database. Here, the returned query is not considered private. The main use-case is generating differentially private machine learning models over private data with MPC [21, 1, 25, 22].

2 Preliminaries

Differential privacy. Differential privacy was introduced in [13] as a technique for quantifying the privacy guarantees of a mechanism. A central concept is the definition of neighbouring datasets which are denoted $x \sim x'$. Intuitively, this definition is used to capture the information we want to protect. Typically x and x' are identical except for the data about one individual. We formally define neighbouring inputs in our setting of the trusted curator in Section 3.1. Differential privacy is a restriction on how much the output distribution of a mechanism can change between any neighbouring input datasets.

▶ **Definition 1** ((ε, δ)-DP). *A randomized mechanism $\mathcal{M}$ satisfies (ε, δ)-differential privacy if for all pairs of neighbouring datasets $x \sim x'$ and all sets of outputs $\mathcal{S}$ we have:*

$$\Pr[\mathcal{M}(x) \in \mathcal{S}] \leq \exp(\varepsilon) \cdot \Pr[\mathcal{M}(x') \in \mathcal{S}] + \delta$$

Multi-party computation. Multi-party computation is a cryptographic technique that allows a set of n mistrustful parties to calculate a function of their own private inputs without revealing them. We consider an MPC protocol based on Shamir secret sharing, where a secret value s is shared by giving each party i the evaluation $f(i)$ of a polynomial f of degree t and coefficients in $\mathbb{F}_p$ such that $f(0) = s$. The protocol assumes an honest majority, i.e., $t < n/2$, and it is actively secure with an abort, meaning that a malicious party deviating from the protocol is caught with overwhelming probability and the honest parties abort the protocol when this happens. In this work, we use Scale-Mamba [3], a framework that implements various MPC protocols in the preprocessing model. In this methodology, the computation has a preprocessing phase where input independent data is generated. This data is then used in the input dependent online phase, where the desired computation over private inputs is performed.

3 Round-differential-privacy in the trusted curator model

In this section, we first formalize the round-based trusted curator model (§3.1). Round-differential-privacy is defined over two distinct notions: (1) Input-differential-privacy (§3.2) protects the honest input, and mirrors the classic notion of differential privacy over a private database. (2) Correlated output differential privacy (§3.3) protects leakage from correlated outputs, and represents a novel privacy notion of general interest for all application domains. Round-differential-privacy (§3.4) is defined over (1) and (2); here, we also formalize multi-round-differential-privacy, to protect the entire honest transcript over multiple interaction rounds in the trusted curator model.

3.1 The trusted curator

We first define our proposed notions of privacy in the "trusted curator" model (Fig. 1), which can then seamlessly be applied to the setting of secure multi-party computation. The trusted curator C interacts with parties $P_1, ..., P_n$, which are assumed to have established private, authenticated communication links with the trusted curator. Interaction proceeds in *rounds*, each consisting of the following phases.

1. **Input phase** All parties send their individual inputs to the trusted curator C, which obtains the input set $x_1, ..., x_n$ from parties $P_1, ..., P_n$ respectively.
2. **Evaluation phase** Upon receiving all inputs, the trusted curator locally computes a known algorithm M over inputs received in the input phase: namely, $\mathbf{y} \leftarrow \mathbf{M}(\mathbf{x})$ where $\mathbf{x} = (x_1, ..., x_n)$ and $\mathbf{y} = (y_1, ..., y_n)$. Further, curator C is assumed to have access to randomness to evaluate randomized algorithms.
3. **Output phase** The trusted curator privately sends each output element y_i in $\mathbf{y}$ to party P_i, and enters the input phase again.

Client corruption. The adversary $\mathcal{A}$ can statically corrupt up to $n-1$ clients, upon which it decides what inputs the corrupted clients submit in each interaction round. The adversary decides corrupted inputs and observes the output for each corrupted client returned from the trusted curator; the adversary cannot corrupt the curator itself. We denote the adversarial output view from a round evaluating mechanism $\mathbf{M}$ on round inputs $\mathbf{x}$ as $\mathbf{M}^{\mathcal{A}}(\mathbf{x})$.

Public outputs. We permit the trusted curator to also return public outputs; these are considered part of the adversarial output view $\mathbf{M}^{\mathcal{A}}(\mathbf{x})$.

Privacy against the network adversary. We assume that the *physical presence* of a party in each round is observable by the network adversary. Since obfuscating the active participation across the network may be challenging, we assume parties to be physically online and to participate in each round, but permit them to submit **dummy inputs**, allowing for passive participation and obfuscating the *logical presence* of a party in a given round. Without dummy inputs, the physical presence of a party will always leak the presence of a logical input contributed by a party to the computation by the trusted curator; in the setting of privacy-preserving markets, for example, the network adversary would learn that a party is submitting some trade in a given round.

Further, we assume that parties can anonymously submit inputs to the trusted curator via techniques such as mixnets [9, 8], thereby hiding their identity from the network adversary. In practice, parties can delegate the physical interaction with the trusted curator model in each round to trusted servers, and only need to come online when they wish to forward a valid, non-dummy input.

Group privacy. Why highlight that *individual* differential privacy guarantees introduced in the subsequent section naturally imply *group privacy*; a mechanism protecting the presence of a single client can do so for multiple clients, consuming equal privacy budget amounts for each additional group member.

3.2 Differential privacy for inputs

In the standard setting of differential privacy, an analyst performs a query on a private database and the result of the query is released to the analyst; a differentially private query bounds how much the analyst output distribution changes, upon adding or removing an entry in the private database.

We adapt the classic notion of differential privacy to the setting of the trusted curator. Instead of protecting private database entries, we first wish to protect inputs submitted by honest clients. We introduce a dummy value that allows clients to have no impact on a given round. Thus, the following definition of neighbouring input vectors follows directly from the standard definition of neighbouring databases under the add/remove relation in the classic setting.

▶ **Definition 2 (neighboring input vectors).** *Let input vectors* $\mathbf{x} = (x_1, ..., x_n)$ *and* $\mathbf{x}' = (x'_1, ..., x'_n)$ *of equal length be such that the following holds true;*

$$\exists i \in [n] : x_j = x'_j \text{ for all } j \neq i \text{ and } (x_i = \textit{dummy} \text{ or } x'_i = \textit{dummy})$$

For a randomized algorithm $\mathbf{M}$ evaluated on input vector $\mathbf{x}$, let $\mathbf{M}^{\mathcal{A}}(\mathbf{x}) = \{Y_j\}_{j \in \mathcal{A}}$ denote output distributions observed by corrupted clients. Then, the following definition follows directly from the standard notion [14] of differential privacy where we consider the input vector as the private database on which the query $\mathbf{M}$ is performed and the adversary obtains the output view $\mathbf{M}^{\mathcal{A}}(\mathbf{x})$ of all corrupted parties. Note that there is no restriction on the output distribution seen only by the honest user.

▶ **Definition 3 ((ε, δ)-input-DP).** *For an evaluation of* (ε, δ)*-input-differentially-private algorithm* $\mathbf{M}$ *in the trusted curator model over neighboring private input vectors* $\mathbf{x} \sim \mathbf{x}'$*, the following must hold for any adversarially observable output event* $\mathcal{S}^{\mathcal{A}}$*.*

$$\Pr[\, \mathbf{M}^{\mathcal{A}}(\mathbf{x}) \in \mathcal{S}^{\mathcal{A}} \,] \leq \exp(\varepsilon) \cdot \Pr[\, \mathbf{M}^{\mathcal{A}}(\mathbf{x}') \in \mathcal{S}^{\mathcal{A}} \,] + \delta$$

As we will see in Section 3.3, input-differential privacy is necessary, but insufficient to protect both in- and output of an honest client in the trusted curator round. Whilst Definition 3 protects the privacy of a user input, it does not not guarantee that the honest output remains private. This motivates *correlated-output*-differential-privacy, introduced in subsequent section Section 3.3. Again, the standard setting of differential privacy does not consider the privacy of the query output, as there is only a single query result which is released publicly or to the adversarial analyst.

▶ **Lemma 4.** *Dark Pools violate* (ε, δ)*-input-differential-privacy for any* $\delta < 1$*.*

Proof. (Sketch) A dark pool venue operator can be idealized as a trusted curator which privately receives trade orders from clients. Upon evaluating the market algorithm in private, it privately outputs trade executions to clients. Assume an honest user submits the only buy order and the corrupted client submitting a sell order observes that its trade order is executed. Any change in the honest counter-party's privately submitted buy order cancels the matching of this order pair, observable to the adversary with probability 1, thereby violating Definition 3. ◀

Adversarially chosen inputs Note that input-differential-privacy in Definition 3 naturally protects against *chosen input attacks*; informally, such an attack permits the adversary to change its inputs and observe induced effects on its output distributions to learn something about honest inputs. However, note that (ε, δ)-input-DP applies equal privacy guarantees to *any input* submitted to the trusted curator. Thus, for appropriately chosen privacy parameters, a chosen input attack on a (ε, δ)-input-DP mechanism will not reveal meaningful information to the adversary, as its chosen input perturbation will not induce a sufficiently observable effect on its output distributions.

3.3 Differential privacy for correlated outputs

In contrast to prior work, where a single output is returned from a differentially-private mechanism, we must protect the privacy of outputs that are returned from trusted curator model to individual clients over private channels. Even if input-differential privacy protects honest inputs, the individual outputs returned to clients may still be strongly correlated, potentially allowing honest outputs to be inferred from corrupted ones.

▶ **Definition 5 ((ε, δ)-correlated-output-DP).** *For an evaluation of (ε, δ)-correlated output differentially private algorithm $\mathbf{M}$ in the trusted curator model over fixed input vector $\mathbf{x}$, the following must hold for any adversarial output event $\mathcal{S}^{\mathcal{A}}$ and any honest output event $\mathcal{S}^{h}$.*

$$\Pr[\, \mathbf{M}^{\mathcal{A}}(\mathbf{x}) \in \mathcal{S}^{\mathcal{A}} \mid \mathbf{M}^{h}(\mathbf{x}) \in \mathcal{S}^{h} \,] \leq \exp(\varepsilon) \cdot \Pr[\, \mathbf{M}^{\mathcal{A}}(\mathbf{x}) \in \mathcal{S}^{\mathcal{A}} \mid \mathbf{M}^{h}(\mathbf{x}) \notin \mathcal{S}^{h} \,] + \delta$$

Definition 5 is interpreted as follows; for any set of inputs and two different honest output events ($\mathbf{M}^{h}(\mathbf{x}) \in \mathcal{S}^{h}$ vs. $\mathbf{M}^{h}(\mathbf{x}) \notin \mathcal{S}^{h}$), the output distribution $\mathbf{M}^{\mathcal{A}}(\mathbf{x})$ of the adversary remains (ε, δ)-similar. In other words, any change in the honest output can only have a bounded effect on the adversarially observable output distribution.

We highlight an immediate consequence of Definition 5 for economic applications; a correlated-output-DP mechanism cannot distribute funds to all clients where the supply of output funds is known or public; an adversary corrupting $n - 1$ clients can trivially infer the funds privately output to the single honest client by just aggregating its own outputs and observing the differnce to the total supply. Thus;

▶ **Lemma 6.** *Economic mechanisms evaluated in the trusted curator model which allocate a fixed supply of "assets" over client outputs violate (ε, δ)-correlated-output-differential privacy for any $\delta < 1$.*

Overcoming this is not straight-forward, as a financial applications cannot be allowed to arbitrarily mint or create funds out of thin air. We overcome these constraints by performing *fuzzy matching* of orders and temporarily freezing funds to achieve correlated-output-differential-privacy in rDP-volume-match (Section 4.1) and rDP-double-auction (Section 4.2).

Applications with correlated outputs. We argue there exist many applications in the trusted curator setting which require correlated outputs; most closely related to this work are economic applications which govern the private allocation of finite resources, which include auctions, markets, financial derivatives and other economic contracts.

3.4 Single-round & Multi-round privacy

Since (ε, δ)-input-DP and (ε, δ)-correlated-output-DP protect different parts of the honest round transcript, we must formally consider two separate privacy budgets which are consumed

with each interaction round in the trusted curator model. We define differential privacy for *each interaction round* with the trusted curator as follows.

▶ **Definition 7 (Round-DP).** *The evaluation of a mechanism that satisfies $(\varepsilon^{in}, \delta^{in})$-input-DP and $(\varepsilon^{out}, \delta^{out})$-correlated-output-DP is $(\varepsilon^{in}, \delta^{in})$-$(\varepsilon^{out}, \delta^{out})$-round-differentially-private.*

Definition 7 implies that the privacy of input and outputs may be parameterized *independently*. Indeed, this permits the trade-off between utility and privacy for different parts of the honest transcript to be decided separately; the input to an evaluation round may require a higher degree of privacy than the returned output or vice versa.

Multi-round privacy When applying differential privacy to queries on a static database, m instances of $(\varepsilon_i, \delta_i)$-differentially private queries taken together are $(\varepsilon_1 + ... + \varepsilon_m, \delta_1 + ... + \delta_m)$ differentially private using basic composition [14] with tighter bounds known using advanced composition [16]. However, in the trusted curator model, the curator accepts fresh inputs in each interaction round, allowing us to consider each round input as disjoint, private data. We define multi-round-differential privacy as the sensitivity of the adversarial output view over m rounds to a change in a single input of a single round $r \in [m]$.

▶ **Definition 8 (m-round-DP).** *Let the adversarial output view over m interaction rounds between n-clients and the trusted curator be given as $\mathbf{M}_1^{\mathcal{A}}(\mathbf{x}_1), ..., \mathbf{M}_m^{\mathcal{A}}(\mathbf{x}_m)$. Then, we define this m-round transcript as;*

$$\mathbf{M}_{mul}^{\mathcal{A}}(\bar{\mathbf{x}}) = (\mathbf{M}_1^{\mathcal{A}}(\mathbf{x}_1), ..., \mathbf{M}_m^{\mathcal{A}}(\mathbf{x}_m)) \;\; \textit{where} \;\; \bar{\mathbf{x}} = (\mathbf{x}_1, ..., \mathbf{x}_m)$$

Multi-round inputs $\bar{\mathbf{x}}$ and $\bar{\mathbf{x}}'$ are neighboring, if there exists unique round $j \in [m]$, such that $\mathbf{x}_j \in \bar{\mathbf{x}}$ and $\mathbf{x}'_j \in \bar{\mathbf{x}}'$ are neighbouring (Definition 2) and $\mathbf{x}_k = \mathbf{x}'_k$ for all other rounds $k \neq j$. Then, we denote an m-round output event for the adversarial and honest client as $\mathcal{S}_{mul}^{\mathcal{A}} = \mathcal{S}_1^{\mathcal{A}}, ..., \mathcal{S}_m^{\mathcal{A}}$ and $\mathcal{S}_{mul}^{h} = \mathcal{S}_1^{h}, ..., \mathcal{S}_m^{h}$ respectively.

The m-round interaction is $(\varepsilon^{in}, \delta^{in})$-$(\varepsilon^{out}, \delta^{out})$-$m$-round differentially private if for neighbouring $\bar{\mathbf{x}}$ and $\bar{\mathbf{x}}'$, any adversarial m-round event $\mathcal{S}_{mul}^{\mathcal{A}}$ and honest m-round event $\mathcal{S}_{mul}^{h}$, the following holds;

$$\begin{aligned} &\Pr[\, \mathbf{M}_{mul}^{\mathcal{A}}(\bar{\mathbf{x}}) \in \mathcal{S}_{mul}^{\mathcal{A}} \,] \\ &\leq \exp(\varepsilon^{in}) \cdot \Pr[\, \mathbf{M}_{mul}^{\mathcal{A}}(\bar{\mathbf{x}}') \in \mathcal{S}_{mul}^{\mathcal{A}} \,] + \delta^{in} \qquad (a) \\ &\forall j \in [m] : \Pr[\, \mathbf{M}_{mul}^{\mathcal{A}}(\bar{\mathbf{x}}) \in \mathcal{S}_{mul}^{\mathcal{A}} \mid \mathbf{M}_j^{h}(\bar{\mathbf{x}}) \in \mathcal{S}_j^{h} \,] \\ &\leq \exp(\varepsilon^{out}) \cdot \Pr[\, \mathbf{M}_{mul}^{\mathcal{A}}(\bar{\mathbf{x}}) \in \mathcal{S}_{mul}^{\mathcal{A}} \mid \mathbf{M}_j^{h}(\bar{\mathbf{x}}) \notin \mathcal{S}_j^{h} \,] + \delta^{out} \qquad (b) \end{aligned}$$

Concretely, m-round-DP bounds the sensitivity of the adversarial output distribution over m-rounds to both (a) a change in the honest users input in the round $j \in [m]$ round and (b) a change in the honest users output in any round $j \in [m]$.

The following theorem relates single-round-DP (def. 7) with m-round-DP (def. 8).

▶ **Theorem 9 (m-round composition).** *Let there be m consecutive interaction rounds with n clients and the trusted curator. In each round, the trusted curator evaluates round-specific algorithms $\mathbf{M}_1, ..., \mathbf{M}_m$ that are $(\varepsilon_1^{in}, \delta_1^{in})$-$(\varepsilon_1^{out}, \delta_1^{out})$, ..., $(\varepsilon_m^{in}, \delta_m^{in})$-$(\varepsilon_m^{out}, \delta_m^{out})$ round-differentially-private and evaluated on round-specific input vectors $\mathbf{x}_1, ..., \mathbf{x}_m$. Then, the m-round evaluation is*

$$(\max_{j \in [m]} \varepsilon_j^{in} \,,\; \max_{j \in [m]} \delta_j^{in}) - (\max_{j \in [m]} \varepsilon_j^{out} \,,\; \max_{j \in [m]} \delta_j^{out})$$

m-round-differentially-private.

Proof. (Theorem 9) In an m-round interaction with n clients and the trusted curator evaluating round-specific algorithms $\mathbf{M}_1, ..., \mathbf{M}_m$, clients provide fresh set of inputs $\mathbf{x}_j$ in each round $j \in [m]$. Definition 8 bounds the sensitivity of the full, adversarial output over m-rounds to any change in (a) honest input or (b) honest output in a single round $j \in [m]$. However, each evaluation of $\mathbf{M}_i$ is independent; in each round, the curator only computes on freshly submitted inputs only. Thus, any change in adversarial output distribution induced by (a) or (b) will be only be observable in round j of the adversarial transcript.

Since $\mathbf{M}_j$ for round $j \in [m]$ is $(\varepsilon_j^{\mathsf{in}}, \delta_j^{\mathsf{in}})$-$(\varepsilon_j^{\mathsf{out}}, \delta_j^{\mathsf{out}})$-round differentially private, it follows that each algorithm $\mathbf{M}_j$ for any round $j \in [m]$ is also

$$(\varepsilon_{\mathsf{max}}^{\mathsf{in}}, \delta_{\mathsf{max}}^{\mathsf{in}})\text{ - }(\varepsilon_{\mathsf{max}}^{\mathsf{out}}, \delta_{\mathsf{max}}^{out}) = (\max_{j\in[m]} \varepsilon_j^{\mathsf{in}}\ ,\ \max_{j\in[m]} \delta_j^{\mathsf{in}})\text{ - }(\max_{j\in[m]} \varepsilon_j^{\mathsf{out}}\ ,\ \max_{j\in[m]} \delta_j^{\mathsf{out}})$$

round-differentially-private. Then, m-round-differential-privacy (Definition 8) is satisfied, since a change to (a) an honest input or (b) output in round $j \in [m]$ will induce an (a) $(\varepsilon_{\mathsf{max}}^{\mathsf{in}}, \delta_{\mathsf{max}}^{\mathsf{in}})$ or (b) $(\varepsilon_{\mathsf{max}}^{\mathsf{out}}, \delta_{\mathsf{max}}^{\mathsf{out}})$ bounded effect on adversarial output $\mathbf{M}_j^{\mathcal{A}}$ in round j only. ◀

4 Round-differentially-private market mechanisms

We propose round-differentially-private market mechanisms in the trusted curator model. These include volume matching of orders, where a batch of buy and sell orders (§4.1) are matched at a given exchange rate determined at an external reference market, and double auctions (§4.2), where buy and sell orders also feature price limits, such that a clearing price must first be computed for each round before orders can be matched. Following a gentle introduction, each algorithm is formally proven to satisfy round-differential-privacy.

To realize any meaningful notion of privacy in practice, we later distribute the trusted curator by means of secure multi-party computation (MPC) in section §5.

4.1 Round-DP volume matching

In volume matching, the exchange rate is pre-determined by an external reference rate. A trader only chooses to submit a sell, buy or to abstain from the round with a dummy order. We introduce a $(\varepsilon^{\mathsf{in}}, \delta^{\mathsf{in}})$-$(\varepsilon^{\mathsf{out}}, \delta^{\mathsf{out}})$-round-differentially-private volume matching algorithm named rDP-Volume-match, overcoming the privacy limitations of the traditional dark pool setting, where a matched buy and sell order pair implies leaking the presence of an order execution to the counter-party and thereby violates both input- and correlated-output differential privacy (Lemmas 4 and 6). We overcome this privacy challenge with the following two phases in the rDP-Volume-match algorithm (Fig. 2).

1. Fuzzy order matching. In the first phase of rDP-volume-match, orders are matched in a "fuzzy" manner; following a preliminary, deterministic matching step which maximizes number of trades, each final trade output (trade/no-trade) for each client is sampled independently from a distribution parameterized by $\varepsilon^{\mathsf{in}}$ and biased towards the preliminary matching result; we adapt this technique from the standard randomized response mechanism [14, 26], that is both $(\varepsilon^{\mathsf{in}}, 0)$-input and $(0, 0)$-correlated output differentially private; the latter property arises naturally from the independent sampling of outputs which occurs in randomized response. Randomized, fuzzy matching of orders also implies that the final aggregate exchange of assets may not sum to zero; in any given round, the total buy volume may not equal the total sell volume. We handle this mismatch in the second phase of rDP-volume-match.

2. Liquidity compensation. We introduce a *liquidity provider*, which compensates for the mismatch between buy and sell volume; however, without any additional treatment, the

adversary corrupting $n-1$ traders *and* the liquidity provider can trivially learn the honest output from the implied flow of assets between corrupt and honest parties (e.g. output correlation). To ensure that the corrupt liquidity provider's output is correlated-output-differentially-private, a randomized amount of its liquidity is frozen; here, the parameterization of rDP-Volume-match permits the choice of an upper limit (ρ^{max}) on frozen volumes of both the risky and numeraire asset types, thereby bounding the opportunity cost imposed on the liquidity provider. We define a *privacy epoch* over multiple rounds in Definition 10, uring which the privacy guarantees of rDP-volume-match hold; if the frozen liquidity is later returned to the liquidity provider, round-differential-privacy is no longer guaranteed. In practice, we argue that it is acceptable to guarantee round-differential-privacy for a bounded number of rounds, during which honest users can complete their multi-round trading strategy without front-running interference. For privacy guarantees to hold indefinitely, assets would have to burned. Note that assets are never minted, preserving the integrity of their supply. We also note that, in principle, multiple liquidity providers could participate in each round of rDP-Volume-Match; we model a single liquidity provider to simplify exposition and formal proofs.

Next, we detail and motivate steps of rDP-volume-match and refer to Fig. 2 for a formal description of the algorithm.

rDP-Volume-match. Following inputs are assumed to be well-formed for simplicity.

- Each trader $\mathcal{P}_i^{\mathsf{trd}} \in (\mathcal{P}_1^{\mathsf{trd}}, ..., \mathcal{P}_n^{\mathsf{trd}})$ inputs trade order $(b_i, s_i, \mathsf{id}_i)$.
- The liquidity provider $\mathcal{P}^{\mathsf{liq}}$ inputs liquidity amounts $(x_0^{\mathsf{liq}}, x_1^{\mathsf{liq}})$.
- The privacy parameters $\varepsilon^{\mathsf{in}}, \varepsilon^{\mathsf{out}}, \delta^{\mathsf{out}} > 0$ and ρ^{max}.

1. Fuzzy order matching

1a. Deterministic matching of buy & sell orders

- Aggregate number of buy and sells; $\forall i$: $B \leftarrow B + b_i$, and $S \leftarrow S + s_i$
- Determine the number of matched pairs; $u \leftarrow \mathsf{min}(B, S)$.
- Match u pairs of buy and sell orders;
  Let $\mathsf{match}_i \in \{0, 1\}$ indicate a match for party P_i for $i \in [n]$.

1b. Randomized response over order matches

- $\forall i$: If match_i, sample $\mathsf{trade}_i \leftarrow\$\, e^{\varepsilon^{\mathsf{in}}}/(1+e^{\varepsilon^{\mathsf{in}}})$, otherwise $\mathsf{trade}_i \leftarrow\$\, 1/(1+e^{\varepsilon^{\mathsf{in}}})$
  - If trade_i is true and $(b_i, s_i) \neq (0, 0)$;
    - Return $(b_i^{\mathsf{out}} = 1, s_i^{\mathsf{out}} = 0, \mathsf{id}_i)$ to $\mathcal{P}_i^{\mathsf{trd}}$ for a buy order $(b_i = 1, s_i = 0)$
    - Return $(b_i^{\mathsf{out}} = 0, s_i^{\mathsf{out}} = 1, \mathsf{id}_i)$ to $\mathcal{P}_i^{\mathsf{trd}} P_i$ for a sell order $(b_i = 0, s_i = 1)$
  - Else return $(b_i^{\mathsf{out}} = 0, s_i^{\mathsf{out}} = 0, \mathsf{id}_i)$ to $\mathcal{P}_i^{\mathsf{trd}}$

2. Liquidity compensation

2a. Liquidity compensation for sampled trades

- Aggregate final buy and sell volumes; $\forall i$: $o^b \leftarrow o^b + b_i^{\mathsf{out}}$, and $o^s \leftarrow o^s + s_i^{\mathsf{out}}$
- Determine the liquidity mismatch; $\Delta_0 \leftarrow (o^s - o^b)$ and $\Delta_1 \leftarrow -\Delta_0$

2b. Randomized liquidity freezing

- Sample frozen volumes; $\forall t \in \{0, 1\} : \rho_t \leftarrow\$\, P_{\mathsf{frz}}$ (Param. by $\rho^{\mathsf{max}}, \varepsilon^{\mathsf{out}}, \delta^{\mathsf{out}}$ in Eq. 4).
- Return liquidity amounts $(y_0^{\mathsf{liq}}, y_1^{\mathsf{liq}}) \leftarrow (x_0^{\mathsf{liq}} + \Delta_0 - \rho_0\,,\, x_1^{\mathsf{liq}} + \Delta_1 - \rho_1)$ to $\mathcal{P}^{\mathsf{liq}}$.
- Freeze (ρ_0, ρ_1), return to $\mathcal{P}^{\mathsf{liq}}$ at the end of the privacy epoch.

Figure 2 RDP-Volume-match algorithm.

Orders in rDP-Volume-match. Let a valid, privately submitted trade order be the tuple (b, s, id), where b and s represent buy and sell bits respectively, and id is the trader identifier. Thus, let $(b, s) \in [(1, 0), (0, 1), (0, 0)]$ represent a buy, sell and dummy order respectively. We fix buy and sell unit volumes such that a single sell and buy order always match in exchanged asset value.

1a. Deterministic matching (1a. in Fig. 2). Let the number of orders sent to the trusted curator by n clients be $\mathbf{x} = \{(b, s, \mathsf{id}_1), ..., (b, s, \mathsf{id}_n)\}$. Then, the maximum possible number of matches between buy $(b, s) = (1, 0)$ and sell $(b, s) = (0, 1)$ orders is computed, which is simply the smaller of the number of buy b and sell s orders. Let the result of the deterministic matching phase be the bit array $\mathsf{match} = (\mathsf{match}_1, ..., \mathsf{match}_n)$, where bit match_i indicates if the i'th submitted order was matched (1) or not (0). Once the total number of preliminary matched pairs is computed, they are assigned randomly to the non-dummy orders; dummy orders are never matched.

1b. Randomized response over matches (1b. in Fig. 2). Here, we apply the standard randomized response mechanism [14, 26] to determine whether a *trade* or *no-trade* is returned to the trader who submitted a valid trade order; for each bit in array match where $\mathsf{match}_i = 1$, the probability of the final trade_i bit equaling 1 or 0 is given by;

$$\begin{aligned} \Pr[\,\mathsf{trade}_i = 1 \,|\, \mathsf{match}_i = 1\,] &= e^{\varepsilon^{\mathsf{in}}}/(1 + e^{\varepsilon^{\mathsf{in}}}) \\ \Pr[\,\mathsf{trade}_i = 0 \,|\, \mathsf{match}_i = 1\,] &= 1/(1 + e^{\varepsilon^{\mathsf{in}}}) \end{aligned} \tag{1}$$

Conversely, for each bit $\mathsf{match}_i = 0$ in match and in the case that party i did not submit a dummy order, the probability of the final trade_i outcome being sampled as 1 or 0 is given by;

$$\begin{aligned} \Pr[\,\mathsf{trade}_i = 1 \,|\, \mathsf{match}_i = 0\,] &= 1/(1 + e^{\varepsilon^{\mathsf{in}}}) \\ \Pr[\,\mathsf{trade}_i = 0 \,|\, \mathsf{match}_i = 0\,] &= e^{\varepsilon^{\mathsf{in}}}/(1 + e^{\varepsilon^{\mathsf{in}}}) \end{aligned} \tag{2}$$

Thus, for parties submitting valid, non-dummy trades, each of the final trading results in array $\mathsf{trade} = [\mathsf{trade}_1, ..., \mathsf{trade}_n]$ is obtained from independently sampling from distributions Eq. 1 or 2 according to the match_i bit output from the deterministic matching subroutine [1a]. Trader outputs are given by the array $[(b_1^{\mathsf{out}}, s_1^{\mathsf{out}}, \mathsf{id}_1), ..., (b_n^{\mathsf{out}}, s_n^{\mathsf{out}}, \mathsf{id}_n)]$, where each entry $(b_i^{\mathsf{out}}, s_i^{\mathsf{out}}, \mathsf{id}_i)$ returned to party i indicates whether a buy $(b_i^{\mathsf{out}}, s_i^{\mathsf{out}}) = (1, 0)$, sell $(b_i^{\mathsf{out}}, s_i^{\mathsf{out}}) = (0, 1)$ or no trade $(b_i^{\mathsf{out}}, s_i^{\mathsf{out}}) = (0, 0)$ was executed;

We emphasize that a trade can only be executed if a non-dummy order was submitted at the beginning of the round, and in the same direction (sell or buy) as intended by the trader. Dummy orders always return $(b^{\mathsf{out}}, s^{\mathsf{out}}) = (0, 0)$ as output; the fuzzy matching is only applied to valid, non-dummy orders only, and thus the trading "interface" remains the same as in traditional volume matching algorithms; a trade order is either filled or not at all.

2a. Liquidity compensation for sampled trades (2a. Fig. 2). Fuzzy matching of orders via randomized response implies that traded volumes from step [1b] in rDP-volume-match do not match precisely; for the trade outputs $[(b_1^{\mathsf{out}}, s_1^{\mathsf{out}}, \mathsf{id}_1), ..., (b_n^{\mathsf{out}}, s_n^{\mathsf{out}}, \mathsf{id}_n)]$, the following can occur;

$$\sum_{i \in [n]} s_i^{\mathsf{out}} \neq \sum_{i \in [n]} b_i^{\mathsf{out}}$$

Since sells and buys may not cancel out, we introduce the presence of a *liquidity provider*, which compensates for this mismatch in traded asset liquidity. Then, the amount of the numeraire asset (Δ_0) and risky asset (Δ_1) provided $(\Delta < 0)$ or received $(\Delta > 0)$ by the

liquidity provider is given as;

$$\Delta_0 = -\sum_{i\in[n]} s_i^{\mathsf{out}} - b_i^{\mathsf{out}} \qquad \Delta_1 = \sum_{i\in[n]} s_i^{\mathsf{out}} - b_i^{\mathsf{out}} \tag{3}$$

The liquidity provider compensates for this liquidity imbalance resulting from fuzzy matching; its initial balances $(x_0^{\mathsf{liq}}, x_1^{\mathsf{liq}})$ are updated to $(x_0^{\mathsf{liq}} + \Delta_0, x_1^{\mathsf{liq}} + \Delta_1)$; however, note that any change in the honest user's trade execution will affect Δ_0, Δ_1 with probability 1, observable by the corrupted liquidity provider and violating correlated-output-differential-privacy (Def. 5); *relaxing* the correlation between the final exchange of assets and the update in funds observed by the liquidity provider can only imply the *minting* or *removal* of funds in the round outputs.

We propose a compromise, which is a randomized mechanism to *freeze liquidity*, protecting the privacy of traders for the m-round duration that the liquidity remains frozen; we call this a privacy epoch (Def. 10). Our algorithm DP-volume-match refrains from minting, preserving the integrity of the underlying asset types.

2b. Randomized liquidity freezing (2b. in Fig. 2). (L8 in Figure 6). The liquidity provider inputs $(x_0^{\mathsf{liq}}, x_1^{\mathsf{liq}})$ amounts of numeraire (0) and risky (1) asset to a given round, and is returned updated reserve balances $(y_0^{\mathsf{liq}}, y_1^{\mathsf{liq}}) = (x_0^{\mathsf{liq}} + \Delta_0 - \rho_0, x_1^{\mathsf{liq}} + \Delta_1 - \rho_1)$, where (ρ_0, ρ_1) is the volume of assets (0) and (1) frozen in the given round and returned at the end of the privacy epoch, chosen to be sufficiently long to protect a common trading strategies executed over multiple rounds.

Note that it would be easy to freeze liquidity with perfect privacy if we had unbounded liquidity. The liquidity provider could provide n units of each asset in every round and liquidity would be frozen such that $(y_0^{\mathsf{liq}}, y_1^{\mathsf{liq}}) = (x_0^{\mathsf{liq}} - n, x_1^{\mathsf{liq}} - n)$. However, the required liquidity would not be feasible for large n. Our mechanism instead provides a trade-off between privacy and frozen liquidity. In each round we sample $\rho_0 \in [0, \rho^{\mathsf{max}}]$ and set $\rho_1 = \rho^{\mathsf{max}} - \rho_0$. We give the probability mass function P_{frz} from which ρ_0 is sampled in Equation (4); this distribution is parameterized by a maximum amount of frozen liquidity $\rho^{\mathsf{max}} \geq 1$ in the round, and correlated-output-differential-privacy parameters $\varepsilon^{\mathsf{out}}, \delta^{\mathsf{out}}$.

$$P_{\mathsf{frz}}(\rho_0) = \begin{cases} \delta^{\mathsf{out}} \cdot \exp(\varepsilon^{\mathsf{out}} \cdot \rho_0) & \rho_0 \in [\ 0 : \lceil \frac{\rho^{\mathsf{max}}-1}{2} \rceil\ ] \\ \delta^{\mathsf{out}} \cdot \exp(\varepsilon^{\mathsf{out}} \cdot (\rho^{\mathsf{max}} - \rho_0)) & \rho_0 \in [\ \lceil \frac{\rho^{\mathsf{max}}-1}{2} \rceil + 1 : \rho^{\mathsf{max}}\ ] \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The sensitivity of $\rho_0 + \Delta_0$ and $\rho_1 + \Delta_1$ to the execution of a single trade is ± 1. Distribution frz allocates probability mass across multiples of unit trade volume; neighbouring freezing events ρ_t and $\rho_t \pm 1$ are allocated probabilities which differ by factor $\exp(\varepsilon^{\mathsf{out}})$. Since we limit the amount of frozen tokens to the range $[0 : \rho^{\mathsf{max}}]$, we must accept a non-zero δ^{out} probability of violating $(\varepsilon^{\mathsf{out}})$-correlated-output-differential-privacy (See Lemma 13).

Parameterization of P_{frz}. The freeze distribution is parameterized by $(\varepsilon^{\mathsf{out}}, \delta^{\mathsf{out}}, \rho^{\mathsf{max}})$, but we note that these cannot be chosen independently; parameters are set so the aggregate probability mass of the P_{frz} is 1. We illustrate various parameterizations of $\varepsilon^{\mathsf{out}}, \delta^{\mathsf{out}}$ and ρ^{max} in Fig. 3. To achieve $(2.5, 4.5 \cdot 10^{-4})$-correlated-output-differential privacy, ρ_{max} must be set to 6, implying that up to 6 unit volumes of each asset type provided by the liquidity provider will be frozen. Lowering the ρ^{max} reduces frozen liquidity, but implies higher privacy parameters δ^{out} or $\varepsilon^{\mathsf{out}}$. For $\rho_{\mathsf{max}} = 6$ and rounds exceeding 10^3 number of submitted orders (as benchmarked in §5.3), we argue the opportunity cost of freezing up to 6 unit volumes of each asset represents an acceptable cost for round-differential-privacy.

Cost of liquidity provisioning. In fuzzy order matching, the worst case liquidity mismatch occurs when all submitted orders are in the equal direction and are all executed or fulfilled. Here, the maximum mismatch in liquidity is exactly the number of clients submitting orders in the round. Thus, the liquidity provider has to provide as much liquidity as number of clients ($x^{\mathsf{liq}}_{0,1} = n$), in addition to ρ_{max} of each asset type in each round. However, in rDP-Volume-match, the exchange rate is decided apriori according to an external reference price; we argue that the vast majority of the liquidity can be sourced directly from the external market trading at the reference price. In the blockchain context, this could be a large Automatic Market Maker with sufficient liquidity, thereby reducing the amount of liquidity required from the liquidity provider to just ρ^{max} of each asset type. We leave the detailed analysis of effective incentivization of liquidity provisioning to future work; we imagine traders submitting trade fees in each round, but do not model this explicitly.

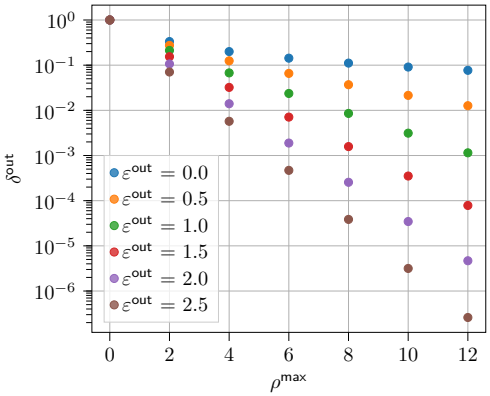

Figure 3 We plot selected parameterizations of P_{frz} in Eq. 4. The choice of parameters represents a trade-off between degree of privacy $(\varepsilon^{\mathsf{out}}, \delta^{\mathsf{out}})$ and frozen funds (ρ^{max}).

▶ **Definition 10 (Privacy epoch).** *We define a privacy epoch over the repeated execution of rDP-Volume-match for m rounds, during which the participating liquidity provider contributes amounts of risky and numeraire assets to be frozen in each round; all frozen funds are returned when the m rounds of the privacy epoch are completed.*

We emphasize that the following privacy properties hold for client in- and outputs during the duration of a single private epoch; once the frozen funds are returned, round-differential-privacy no longer holds. For purposes of mitigating front-running, we argue that the epoch duration should be chosen to be sufficiently long permit the execution of common, long-running honest user strategies. Alternatively, if the frozen funds provided by the liquidity provider are never returned or burnt, the following privacy properties hold absolutely.

We refer to Appendix A for formal proofs of the following theorem and lemmas which demonstrate round-differential-privacy for rDP-Volume-matching.

▶ **Theorem 11.** *rDP-Volume-matching is $(\varepsilon^{in} + \varepsilon^{out}, \delta^{out})$-$(\varepsilon^{out}, \delta^{out})$-$m$-round-differentially-private.*

Theorem 11 follows directly from Lemmas 13 and 14, while the latter is demonstrated by leveraging bounds from Lemmas 12 and 13; we refer to the proof strategy in Appendix A.

▶ **Lemma 12.** *rDP-Volume-matching is $(\varepsilon^{in}, 0)$-input differentially private against an adversary that sees the adversarial trade outputs.*

▶ **Lemma 13.** *rDP-Volume-matching is $(\varepsilon^{out}, \delta^{out})$-correlated-output-differentially-private.*

▶ **Lemma 14.** *rDP-Volume-matching is $(\varepsilon^{in} + \varepsilon^{out}, \delta^{out})$-input-differentially-private.*

4.2 Round-DP double auctions

We propose a round-differentially-private double auction algorithm, called DP-double-auction for the trusted curator setting. Here, we introduce an initial sub-routine to compute a $(\varepsilon^{\mathsf{in}}_1, 0)$-input-differentially-private *clearing price* from trade orders input to the round. Subsequently,

rDP-volume-match (§4.1) is performed on the subset of trade orders with price limits consistent with the clearing price.

For each round, we assume a discrete price range $\mathbf{r} = [r_1, ..., r_l]$; without differentially privacy, the discrete price maximizing the number of order matches would be selected; to preserve input-differential-privacy, the exponential mechanism is applied to determine the clearing price.

Orders in rDP-double-auction. Inputs submitted to DP-double-auction are input in the form of $\mathbf{x} = [(\mathbf{w}_1, \mathsf{dir}_1, \mathsf{id}_1), ..., (\mathbf{w}_n, \mathsf{dir}_n, \mathsf{id}_n)]$, where each valid trade order $(\mathbf{w}_i, \mathsf{dir}_i, \mathsf{id}_i)$, contains bit array $\mathbf{w}_i = [w_{i1}, ..., w_{il}]$, where each bit w_{ij} indicates whether user i is willing to buy ($\mathsf{dir}_i = 0$) or sell ($\mathsf{dir}_i = 1$) at price $r_j \in \mathbf{r}$.

Round-differentially-private clearing price. In the spirit of the round-differentially-private mechanisms introduced thus far, rDP-double-auction first computes a deterministic clearing price, and then applies a randomized, mechanism to determine the final, $(\varepsilon_1^{\mathsf{in}}, 0)$-input-differentially private clearing price. Note that since the clearing price is "publicly" released, there is no private client output privacy to protect. For each discrete price index $j \in [l]$, we aggregate trade orders willing to sell or buy at price $r_j \in \mathbf{r}$. Let S_j denote all sell orders willing to sell at price $r_j \in [\mathbf{r}]$ and B_j denote all buy orders willing to buy at price $r_j \in [\mathbf{r}]$. Then, the number of matched pairs at price r_j is given by $u_j = \min(B_j, S_j)$, resulting in $\mathbf{u_x} = \{u_1, ..., u_j\}$. Here, we interpret $(\mathbf{u_x} : \mathbb{Z}^{\leq l} \to \mathbb{Z}^{\leq n/2})$ as the *utility function* for sampling the final clearing price with the exponential mechanism.

The *exponential mechanism*, first introduced by McSherry and Talwar [17], realizes a probability distribution over a range of events, for which the mechanism designer can express a *utility* score function $\mathbf{u_x}$ applicable to each event; thus, events can be allocated probability mass proportional to $\exp\left(\varepsilon_1^{\mathsf{in}} \cdot \mathbf{u_x}(r)/(2\Delta u)\right)$, where Δu denotes the sensitivity of the utility function to a change to a single input. In other words, the mechanism designer can influence the probability distribution over events by allocating higher utility scores to preferred outcomes.

In DP-double-auction, the senstivity of $\mathbf{u}_x(j)$ at any discrete price index $j \in [l]$ is simply 1; thus $\Delta u = 1$. A change in an honest input from valid to a dummy order or from a dummy order to valid affects at most one match per price. Therefore,

$$\Delta \mathbf{u} = \max_{\mathbf{x}, \mathbf{x}'} \max_{j \in [l]} \mid \mathbf{u_x}(j) - \mathbf{u_{x'}}(j) \mid \leq 1$$

Then, the probability distribution over which the clearing price is sampled is given by the exponential mechanism parameterized by utility function $\mathbf{u_x}$, which in turn is determined from the submitted trade orders $\mathbf{x}$. Thus, the probability of each discrete price $r_j \in \mathbf{r}$ is given by;

$$\Pr[j] = \frac{\exp(\varepsilon_1^{\mathsf{in}} \cdot \mathbf{u_x}(j)/2)}{\sum_{i \in [|\mathbf{r}|]} \exp(\varepsilon_1^{\mathsf{in}} \cdot \mathbf{u_x}(i)/2)} \tag{5}$$

Since the exponential mechanism is $(\varepsilon_1^{\mathsf{in}}, 0)$-input-differentially-private over all inputs ([17]), we consume $\varepsilon_1^{\mathsf{in}}$ of our $(\varepsilon^{\mathsf{in}}, 0)$-input-differential-privacy budget when outputting the clearing price computed over $\mathbf{x}$, leaving another $\varepsilon_2^{\mathsf{in}}$ for the subsequent rDP-volume-match at price r, such that $\varepsilon^{\mathsf{in}} = \varepsilon_1^{\mathsf{in}} + \varepsilon_2^{\mathsf{in}}$.

rDP-Volume matching at clearing price. We subsequently apply rDP-volume-match from Section 4.1 at sampled clearing price from the preceding exponential mechanism step, returning the trade outputs from DP-volume-match privately to each trading client and frozen liquidity amounts to the liquidity provider.

▶ **Theorem 15.** *rDP-double auction is m-round-differentially-private.*

We refer to Appendix A for the formal proof of Theorem 15.

5 Round-DP market mechanisms with MPC

To obtain a fair market in practice, we instantiate the trusted curator with a set of MPC parties who execute the market mechanisms described in Sections 4.1 and 4.2 using an MPC protocol. In Sections 5.1 and 5.2, we present a formal description of the proposed market mechanisms, as well as some textual explanation for the steps of the algorithms where some care is required to ensure the efficiency of the MPC execution.

We implement our oblivious algorithms in the online/offline preprocessing paradigm; during a preceding (off-line) preprocessing phase, secret-shared data is generated which is *independent* of secret client inputs. When running experiments, we focus on benchmarking online phases, as pre-processing phase can be outsourced or parallelized. Thus, in Section 5.3, we present online runtimes for both the rDP-volume-match algorithm and the rDP-double-auction algorithm. We show that, even though these algorithms satisfy stronger privacy guarantees than those in previous works on dark pools using MPC, high order throughput is still achieved. Indeed, the runtimes of our rDP-volume-match algorithm are in the same order of magnitude as those of the Bucket Match algorithm from [11], which provides a similar functionality but without round-differential-privacy. The rDP-double-auction algorithm, on the other hand, has a more expensive input correctness check, becoming more expensive as we consider more price points. Even so, it can process around one thousand orders in just 2 seconds when considering 100 price points. Thus, we introduce the possibility for users to choose the prices at which they wish to trade while achieving practical throughput and satisfying round-differential-privacy.

5.1 rDP-volume-match with MPC

The formal description of the rDP-volume-match algorithm instantiated with MPC is presented in Figures 5 and 6. Note that both the order format and the InputCheckVM procedure in Figure 5, as well as the first 2 steps of the MatchVol procedure in Figure 6 are identical to the ones in the Bucket Match algorithm from [11].

Randomness sampling. We note that distributions sampled during randomized matching response and the liquidity compensation are independent of the input orders, and can thus be obliviously sampled ahead of time by running the procedure NoiseGen in Figure 6 together during the preprocessing phase of the MPC algorithm. The sampling, described in Figure 4, is performed using the inverse transform sampling method, adapted from [15]. To sample a random value from a distribution given by probability mass function P, we start by taking the corresponding cumulative distribution function, $F_X(x) = \sum_{x_i \leq x} P(X = x_i)$. We then sample a secret shared value $\langle z \rangle \in (0, 1]$ uniformly at random, which can be derived from a randomly generated integer as shown in [15]. The desired distribution can now be obtained by taking the first x such that $F_X(x)$ is greater or equal to $\langle z \rangle$.

Input format check. Since the orders are secret shared among the MPC parties, a format verification step is required. I.e., we need to check that every order i is such that $(b_i, s_i) \in \{(1, 0), (0, 1), (0, 0)\}$. To do so, we run the InputCheckVM procedure in Figure 5, where orders without the correct format are rejected.

Fuzzy order matching. To achieve the desired differential privacy guarantees, we avoid revealing any of the secret shared values throughout the computation. This is unlike the

Sample: On input P, the probability distribution of a discrete random variable X that may take k different values $x_1, ...x_k$:

1. Sample $\langle z \rangle \in (0, 1]$ uniformly at random.
2. For all i: $F_i \leftarrow \sum_{j=1}^{i} P(X = x_j)$
3. For all i: $\langle c_i \rangle \leftarrow (F_i \geq \langle z \rangle)$.
4. For all i: $c_i \leftarrow \mathsf{Open}(\langle c_i \rangle)$.
5. Return x_j for the lowest j such that $c_j = 1$.

Figure 4 Sampling from a probability distribution P with MPC.

InputCheckVM: On input $\mathbf{x}' = [x'_1, ..., x'_n]$, where $x'_i = (\langle b_i \rangle, \langle s_i \rangle, \langle \mathsf{id}_i \rangle)$ and $b_i, s_i, \mathsf{id}_i \in \mathbb{F}_p$:

`Check validity of inputs bits:` $(0,0) \vee (0,1) \vee (1,0)$.

1. Sample $\alpha_i, \beta_i, \gamma_i$ uniformly at random.
2. $\langle t_i \rangle \leftarrow \alpha_i \cdot (\langle b_i \rangle \cdot \langle b_i \rangle - \langle b_i \rangle) + \beta_i \cdot (\langle s_i \rangle \cdot \langle s_i \rangle - \langle s_i \rangle) + \gamma_i \cdot (\langle b_i \rangle \cdot \langle s_i \rangle)$
3. $t_i \leftarrow \mathsf{Open}(\langle t_i \rangle)$
4. If $t_i = 0$ then add x'_i to a list $\mathbf{x}$, otherwise reject x'_i.
5. Return $\mathbf{x}$.

Figure 5 Input correctness check for the rDP-volume-match algorithm (from [11], Figure 3).

Bucket Match mechanism from [11], where the trading direction with the most total volume was revealed, and the matching procedure was simplified by opening successful orders as soon as they were matched. As a consequence, we obtain a more complex, oblivious procedure, described in MatchVol in Figure 6. Here, we calculate the cumulative total volume for each i and for each direction, thus obtaining $\langle \sigma_i^b \rangle$ and $\langle \sigma_i^s \rangle$ (note that we need to perform the calculations in both directions to hide which direction has more total volume). We then compare $\langle u \rangle$ (the total matched volume in each direction) with the cumulative volume at each index i, and accept every order i until $\langle u \rangle$ is exceeded. A randomized response over the matches is obtained by using the randomness $\langle \pi_i \rangle$ sampled during MPC pre-processing.

Liquidity compensation. This phase of the oblivious algorithm, realized in step 12 of the MatchVol procedure (Fig. 6), is identical to the liquidity compensation procedure described in Section 4.1, except that we are now operating over secret shared values.

5.2 rDP-double-auction with MPC

The formal description of the rDP-double-auction algorithm instantiated with MPC is presented in Figures 7 and 8.

Input format check. The correctness of the inputs is verified by the procedure InputCheckDA in Figure 7, which checks that $\langle \mathsf{dir}_i \rangle$ as well as every $\langle w_{ij} \rangle$ are bits and rejects order i if that is not the case. For the orders in the correct format, this procedure additionally converts $\langle w_{ij} \rangle$ and $\langle \mathsf{dir}_i \rangle$ into a sequence of pairs $(\langle b_{ij} \rangle, \langle s_{ij} \rangle)$, where $\langle b_{ij} \rangle$ and $\langle s_{ij} \rangle$ represents whether order i is a buy, sell or a dummy for price j (i.e., $\langle b_{ij} \rangle$ and $\langle s_{ij} \rangle$ have the same meaning as in Section 5.1, but are now associated with a specific price r_j).

NoiseGen: Use Sample from Figure 4 to compute the noise for steps 5 and 9:

- For all i: $\langle \pi_i \rangle \leftarrow \mathsf{Sample}(P_{\mathsf{rr}})$ (def. in Eq. 1).
- $\langle \rho_0 \rangle, \langle \rho_1 \rangle \leftarrow \mathsf{Sample}(P_{\mathsf{frz}})$ (def. in Eq. 4)

rDP-volume-matching: On input $\mathbf{x}'$, $\mathbf{x}^{\mathsf{liq}}$, submitted by $(\mathcal{P}_1^{\mathsf{trd}}, ..., \mathcal{P}_n^{\mathsf{trd}})$ and $\mathcal{P}^{\mathsf{liq}}$, respectively, where $\mathbf{x}' = [x'_1, ..., x'_n]$, $x'_i = (\langle b_i \rangle, \langle s_i \rangle, \langle \mathsf{id}_i \rangle)$, $\mathbf{x}^{\mathsf{liq}} = (\langle x_0^{\mathsf{liq}} \rangle, \langle x_1^{\mathsf{liq}} \rangle)$ and $b_i, s_i, \mathsf{id}_i, x_0^{\mathsf{liq}}, x_1^{\mathsf{liq}} \in \mathbb{F}_p$:

1. Let $\mathbf{x} \leftarrow \mathsf{InputCheckVM}(\mathbf{x}')$
2. Let $\mathbf{y}, \mathbf{y}^{\mathsf{liq}} \leftarrow \mathsf{MatchVol}(\mathbf{x}, \mathbf{x}^{\mathsf{liq}})$
3. Return $\mathbf{y} = [y_1, ..., y_n]$, $\mathbf{y}^{\mathsf{liq}}$ to $(\mathcal{P}_1^{\mathsf{trd}}, ..., \mathcal{P}_n^{\mathsf{trd}})$ and $\mathcal{P}^{\mathsf{liq}}$, respectively.

Subroutines invoked by rDP-volume-matching

MatchVol: On input $\mathbf{x} = [x_1, ..., x_n]$ and $\mathbf{x}^{\mathsf{liq}} = (\langle x_0^{\mathsf{liq}} \rangle, \langle x_1^{\mathsf{liq}} \rangle)$:

Step [1a] Deterministic matching of buy & sell orders

1. For all i: $\langle B \rangle \leftarrow \langle B \rangle + \langle b_i \rangle$, and $\langle S \rangle \leftarrow \langle S \rangle + \langle s_i \rangle$
2. Let $\langle c \rangle \leftarrow (\langle S \rangle > \langle B \rangle)$ and $\langle u \rangle \leftarrow \langle c \rangle \cdot \langle B \rangle + (1 - \langle c \rangle) \cdot \langle S \rangle$.
3. For all i: $\langle \mathsf{big}_i \rangle \leftarrow \langle c \rangle \cdot \langle s_i \rangle + (1 - \langle c \rangle) \cdot \langle b_i \rangle$.
4. For all i, let $\langle \sigma_i^b \rangle \leftarrow \sum_{h=1}^{i} \langle b_h \rangle$ and $\langle \sigma_i^s \rangle \leftarrow \sum_{h=1}^{i} \langle s_h \rangle$.
5. For all i, let $\langle \sigma'_i \rangle \leftarrow \langle c \rangle \cdot \langle \sigma_i^s \rangle + (1 - \langle c \rangle) \cdot \langle \sigma_i^b \rangle$
6. For all i, let $\langle \mathsf{match}'_i \rangle \leftarrow (\langle \sigma'_i \rangle \leq \langle u \rangle) \cdot \langle \mathsf{big}_i \rangle$
7. For all i: $\langle \mathsf{match}_i \rangle \leftarrow (1 - \langle c \rangle) \cdot \langle s_i \rangle + \langle c \rangle \cdot \langle b_i \rangle + \langle \mathsf{match}'_i \rangle$
8. Set $\mathsf{match} = [\langle \mathsf{match}_1 \rangle, ..., \langle \mathsf{match}_n \rangle]$

Step [1b] Randomized response over order matches

9. For all i:
   - Let $\langle \mathsf{trade}_i \rangle \leftarrow \langle \pi_i \rangle \cdot \langle \mathsf{match}_i \rangle + (1 - \langle \pi_i \rangle) \cdot (1 - \langle \mathsf{match}_i \rangle)$
   - Let $\langle b_i^{\mathsf{out}} \rangle \leftarrow \langle b_i \rangle \cdot \langle \mathsf{trade}_i \rangle$
   - Let $\langle s_i^{\mathsf{out}} \rangle \leftarrow \langle s_i \rangle \cdot \langle \mathsf{trade}_i \rangle$
   - Add $y_i = (\langle b_i^{\mathsf{out}} \rangle, \langle s_i^{\mathsf{out}} \rangle, \langle \mathsf{id}_i \rangle)$ to the output list $\mathbf{y}$.

Step [2a] Liquidity compensation for sampled trades

10. For all i: $\langle o^b \rangle \leftarrow \langle o^b \rangle + \langle b_i^{\mathsf{out}} \rangle$, and $\langle o^s \rangle \leftarrow \langle o^s \rangle + \langle s_i^{\mathsf{out}} \rangle$
11. Let $\langle \Delta_0 \rangle \leftarrow (\langle o^s \rangle - \langle o^b \rangle)$ and $\langle \Delta_1 \rangle \leftarrow -\langle \Delta_0 \rangle$

Step [2b] Randomized liquidity freezing

12. $\mathbf{y}^{\mathsf{liq}} = (\langle y_0^{\mathsf{liq}} \rangle, \langle y_1^{\mathsf{liq}} \rangle) \leftarrow (\langle x_0^{\mathsf{liq}} \rangle + \langle \Delta_0 \rangle - \langle \rho_0 \rangle\,,\ \langle x_1^{\mathsf{liq}} \rangle + \langle \Delta_1 \rangle - \langle \rho_1 \rangle)$
13. Update $(\langle \rho_0^{\mathsf{frz}} \rangle, \langle \rho_1^{\mathsf{frz}} \rangle) \leftarrow (\langle \rho_0^{\mathsf{frz}} \rangle, \langle \rho_1^{\mathsf{frz}} \rangle) + (\langle \rho_0 \rangle, \langle \rho_1 \rangle)$
14. Return $\mathbf{y} = [y_1, ..., y_n]$, $\mathbf{y}^{\mathsf{liq}}$.

Figure 6 rDP-volume-matching algorithm with MPC

InputCheckDA: On input $\mathbf{x}' = [x'_1, ..., x'_n]$, where $x'_i = (\mathbf{w}_i, \langle \mathsf{dir}_i \rangle, \langle \mathsf{id}_i \rangle)$, $\mathbf{w}_i = [\langle w_{i1} \rangle, ..., \langle w_{il} \rangle]$ and $w_{ij}, \mathsf{dir}_i, \mathsf{id}_i \in \mathbb{F}_p$:

Check all inputs are bits.

1. For all j: sample α_{ij} uniformly at random.
2. Sample β_i uniformly at random.
3. $\langle t_i \rangle \leftarrow \alpha_{i1} \cdot (\langle w_{i1} \rangle \cdot \langle w_{i1} \rangle - \langle w_{i1} \rangle) + ... + \alpha_{il} \cdot (\langle w_{il} \rangle \cdot \langle w_{il} \rangle - \langle w_{il} \rangle)$
4. $\langle t_i \rangle \leftarrow \langle t_i \rangle + \beta_i \cdot (\langle \mathsf{dir}_i \rangle \cdot \langle \mathsf{dir}_i \rangle - \langle \mathsf{dir}_i \rangle)$
5. $t_i \leftarrow \mathsf{Open}(\langle t_i \rangle)$
6. If $t_i \neq 0$ then reject $\mathbf{x}'_i$. Otherwise, continue to the next step.
7. For all j, let $\langle b_{ij} \rangle = \langle w_{ij} \rangle \cdot (1 - \langle \mathsf{dir}_i \rangle)$ and $\langle s_{ij} \rangle = \langle w_{ij} \rangle \cdot \langle \mathsf{dir}_i \rangle$.
8. Add $x_i = (\langle b_{i1} \rangle, \langle s_{i1} \rangle, ..., \langle b_{il} \rangle, \langle s_{il} \rangle, \langle \mathsf{id}_i \rangle)$ to a list $\mathbf{x}$.
9. Return $\mathbf{x}$.

Figure 7 Input correctness check for rDP-double-auction.

Exponential mechanism. We obliviously determine how many orders can be matched at each price point by calculating $\langle u_j \rangle$ for each price r_j the same way as $\langle u \rangle$ was calculated in the rDP-volume-matching algorithm. The exponential mechanism can now be used to select the best trading price. The probability $\Pr[j]$ associated with price point r_j depends on the corresponding utility value $\langle u_j \rangle$, and since the utility must remain private, the calculated probabilities $\Pr[j]$ will also be secret shared values. While this does not affect the sampling procedure (the algorithm in Figure 4 remains unchanged if the probability mass function is private), computing each $\Pr[j]$ will require the expensive evaluation of a secure exponentiation.

To avoid exponentiation and efficiently compute the selection probabilities with MPC, we use the techniques proposed in [5]. Firstly, instead of considering the selection probabilities as given in Eq. 5, we reduce the complexity by calculating unnormalized probabilities $\langle W_j \rangle$ (called *weights*), where $\langle W_j \rangle = \exp(\varepsilon \cdot \langle u_j \rangle / 2)$. The clearing price sampling can later be performed using these weights by multiplying $\langle z \rangle$ with $\sum_{j=1}^{l} \exp(\varepsilon \cdot \langle u_j \rangle / 2)$, as shown in steps 4-8 of FindPrice in Figure 8. Secondly, there are two possible solutions for computing the weights according to the value of $\varepsilon_1^{\mathsf{in}}$ (recall that $\varepsilon_1^{\mathsf{in}}$ is the input privacy budget consumed when executing the exponential mechanism; $\varepsilon_1^{\mathsf{in}}$ is public and fixed beforehand):

(i) For $\varepsilon_1^{\mathsf{in}} = 2 \cdot \ln(2)$, we get $\langle W_j \rangle = 2^{\langle u_j \rangle}$. This value can be directly written as $(\langle 0 \rangle, \langle 0 \rangle, \langle 2 \rangle, \langle u_j \rangle)$ by using the floating-point notation introduced in [2]. With this notation, a secret shared floating-point value $\langle f \rangle$ is represented as a tuple $(\langle s \rangle, \langle o \rangle, \langle v \rangle, \langle p \rangle)$ with $f = (1 - 2 \cdot s) \cdot (1 - o) \cdot v \cdot 2^p$, where s is the sign bit (set to 1 when f is negative), o is the zero bit (set to 1 when f is zero), v is the mantissa and p the exponent.

(ii) For $\varepsilon_1^{\mathsf{in}} = 2 \cdot \ln(2)/2^d$, where $d \in \mathbb{N}$, we get $\langle W_j \rangle = 2^{\langle u_j \rangle / 2^d} = 2^{\lfloor \langle u_j \rangle / 2^d \rfloor} \cdot 2^{(\langle u_j \rangle \mod 2^d)/2^d}$. The weight $\langle W_j \rangle$ can thus be obtained by calculating the exponentiation with base 2 on the integer part of $\langle u_j \rangle / 2^d$, and multiplying it by a corrective term $2^{(\langle u_j \rangle \mod 2^d)/2^d}$ which takes one out of 2^d possible values depending on u_j. The 2^d possible terms are publicly pre-computed and the correct one is obliviously selected using $\langle u_j \rangle$.

There is an additional procedure in [5] for calculating the weights for arbitrary values of $\varepsilon_1^{\mathsf{in}}$. This procedure relies on the decomposability of the considered utility function, meaning that clients can locally calculate the weights associated with their own inputs and these can later be combined to obtain a correct global weight using MPC. Since our utility function is not decomposable, this method is not applicable. An alternative for computing the weights for arbitrary $\varepsilon_1^{\mathsf{in}}$ would be to first publicly calculate all the possible weights according to the amount of submitted orders, and then obliviously select the correct weight for each price point. This would however imply several secure comparisons and become inefficient for large amounts of submitted orders and available price points. It is therefore preferable to choose $\varepsilon_1^{\mathsf{in}}$ according to the formats in (i) or (ii), which already provide considerable flexibility.

After a price is selected, we can run the rDP-volume-matching algorithm in Figure 6, starting from step 3 of the MatchVol procedure. Note that since we do not know which orders accept the selected price, every order submitted to the double auction will also be considered when subsequently executing the volume matching. Orders that did not accept the select price will appear as dummies during the matching.

5.3 Experiments

To benchmark the performance of our MPC algorithms, we implemented and executed them using Scale-Mamba [3] with Shamir secret sharing between 3 parties. All the parties are run on identical machines with an Intel i-9900 CPU and 128GB of RAM. The ping time between

rDP-double-auction: On input $\mathbf{x}'$, $\mathbf{x}^{\mathsf{liq}}$, submitted by $(\mathcal{P}_1^{\mathsf{trd}}, ..., \mathcal{P}_n^{\mathsf{trd}})$ and $\mathcal{P}^{\mathsf{liq}}$, respectively, where $\mathbf{x}' = [x'_1, ..., x'_n]$, $x'_i = (\mathbf{w}_i, \langle \mathsf{dir}_i \rangle, \langle \mathsf{id}_i \rangle)$, $\mathbf{w}_i = [\langle w_{i1} \rangle, ..., \langle w_{il} \rangle]$, $\mathbf{x}^{\mathsf{liq}} = (\langle x_0^{\mathsf{liq}} \rangle, \langle x_1^{\mathsf{liq}} \rangle)$ and $w_{ij}, \mathsf{dir}_i, \mathsf{id}_i, x_0^{\mathsf{liq}}, x_1^{\mathsf{liq}} \in \mathbb{F}_p$, as well as a list of prices $\mathbf{r} = [r_1, ..., r_l]$:

1. Let $\mathbf{x} \leftarrow \mathsf{InputCheckDA}(\mathbf{x}')$
2. $\mathbf{x}^{\mathsf{match}}, \langle c_R \rangle, \langle u_R \rangle \leftarrow \mathsf{FindPrice}(\mathbf{x})$
3. Execute MatchVol from Figure 6 from step 3 with inputs $\mathbf{x}^{\mathsf{match}} = [x_1^{\mathsf{match}}, ..., x_n^{\mathsf{match}}]$, $\mathbf{x}^{\mathsf{liq}}$, $\langle c_R \rangle$ and $\langle u_R \rangle$.

Subroutine invoked by rDP-double-auction

FindPrice: On input $\mathbf{x} = [x_1, ..., x_n]$, where $x_i = (\langle b_{i1} \rangle, \langle s_{i1} \rangle, ..., \langle b_{il} \rangle, \langle s_{il} \rangle, \langle \mathsf{id}_i \rangle)$:

1. For all j: $\langle B_j \rangle \leftarrow \langle B_j \rangle + \langle b_{1j} \rangle + ... + \langle b_{nj} \rangle$, and $\langle S_j \rangle \leftarrow \langle S_j \rangle + \langle s_{1j} \rangle + ... + \langle s_{nj} \rangle$.
2. For all j, let $\langle c_j \rangle \leftarrow (\langle S_j \rangle > \langle B_j \rangle)$ and $\langle u_j \rangle \leftarrow \langle c_j \rangle \cdot \langle B_j \rangle + (1 - \langle c_j \rangle) \cdot \langle S_j \rangle$.
3. Calculate weights $\langle W_1 \rangle, ..., \langle W_l \rangle$ using Algorithm 3 from [5] on input $\langle u_1 \rangle, ..., \langle u_l \rangle$.
4. For all j: $\langle F_j \rangle \leftarrow \sum_{h=1}^{j} \langle W_h \rangle$
5. Sample $\langle z' \rangle \in (0, 1]$ uniformly at random and let $\langle z \rangle \leftarrow \langle z' \rangle \cdot \langle F_l \rangle$.
6. For all j: $\langle q_j \rangle \leftarrow (\langle F_j \rangle \geq \langle z \rangle)$.
7. For all j: $q_j \leftarrow \mathsf{Open}(\langle q_j \rangle)$.
8. $R \leftarrow r_j$ for the lowest j such that $q_j = 1$.
9. Set $\mathbf{x}_i^{\mathsf{match}} = [\langle b_{iR} \rangle, \langle s_{iR} \rangle, \langle \mathsf{id}_i \rangle]$.
10. Return $\mathbf{x}^{\mathsf{match}} = [x_1^{\mathsf{match}}, ..., x_n^{\mathsf{match}}]$, $\langle c_R \rangle$ and $\langle u_R \rangle$.

Figure 8 rDP-double-auction algorithm with MPC

all the machines is 1.003 ms. Precise numerical values for the results presented here are given in Appendix B.

Online phase of rDP-volume-match. The runtimes for the online phase of one round of the rDP-volume-match algorithm for an increasing number of submitted orders can be found in Figure 9. These runtimes include the InputCheckVM procedure described in Figure 5 which, is identical to the one in the "Bucket Match" dark pool algorithm from [11], and has an average runtime of 0.00013 seconds (0.13 ms) per order. The randomness sampling for the randomized matching response and the frozen liquidity can be done in the preprocessing phase of the MPC, and is thus not considered for the presented results. The runtimes increase approximately linearly with the number of orders (note the logarithmic scale on the horizontal axis), and we see that our algorithm achieves a high order throughput, taking just under 4 seconds to process 10 thousand orders. While InputCheckVM is identical to the input correctness check of the Bucket Match algorithm from [11], our matching procedure MatchVol is slower than the one in Bucket Match. Because of the stronger privacy guarantees we want to satisfy, we cannot reveal intermediary computation results such as which direction has larger total volume or which orders are already matched, as in the Bucket Match. This results in a more complex (and thus more expensive) matching phase. Nonetheless, we note that our runtimes are in the same order of magnitude as the ones presented in [11]. Our rDP-volume-match algorithm can

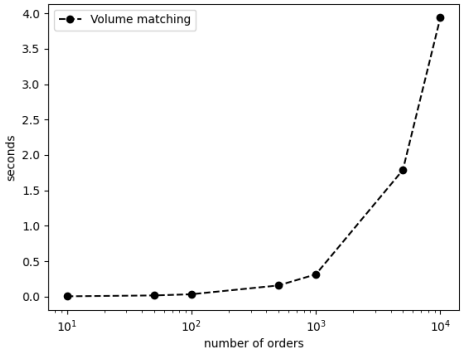

Figure 9 Runtimes in seconds (with logarithmic scale on the x-axis) for the online phase of the rDP-volume-match algorithm.

process 10 thousand orders in 3.94 seconds. The Bucket Match algorithm, on the other hand, processes 26838 orders in 4.14 seconds, i.e., it processes around 2.5 times as many orders in a similar amount of time. The slightly lower throughput of rDP-volume-match should still be high enough for most real-world applications, especially considering the improved privacy it provides.

Online phase of rDP-double-auction. The runtimes for the online phase of the rDP-double-auction algorithm for an increasing number of submitted orders and different values of $\varepsilon_1^{\text{in}}$ can be found in Figure 9. These runtimes include the InputCheckDA procedure described in Figure 7, as well as the FindPrice procedure from Figure 8 and the MatchVol from Figure 6 starting from step 3.

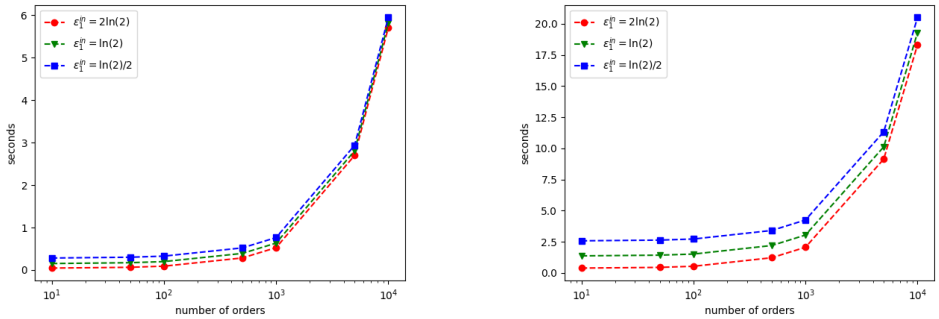

Figure 10 Runtimes in seconds (with logarithmic scale on the x-axis) for the online phase of the rDP-double-auction algorithm with different values of $\varepsilon_1^{\text{in}}$, showing: (left) selection between 10 different price points; (right) selection between 100 different price points. $\varepsilon_1^{\text{in}}$ is the amount of input privacy budget consumed when executing the exponential mechanism to find the clearing price.

The average runtime of InputCheckDA is of 0.00030 seconds (0.30 ms) per order when considering 10 price points and 0.00145 seconds (1.45 ms) per order when considering 100 price points. The percent contribution of this part of the algorithm to the total runtime becomes more significant as the number of orders increases, constituting around 50% of the total runtime across all $\varepsilon_1^{\text{in}}$ values when considering 10 thousand orders with 10 price points, and 70% to 80% when considering 10 thousand orders with 100 price points, depending on the choice of $\varepsilon_1^{\text{in}}$. The FindPrice procedure, on the other hand, does not get significantly slower with the increase in the number of orders. This is also the only part of the algorithm that depends on the choice of $\varepsilon_1^{\text{in}}$, since the method for calculating the weights associated with each price point changes depending on $\varepsilon_1^{\text{in}}$, as described in Section 5.2. As expected, the difference in runtime for different $\varepsilon_1^{\text{in}}$'s becomes more noticeable when considering more price points, with FindPrice taking around 2.2 seconds more with $\varepsilon_1^{\text{in}} = \ln(2)/2$ than with $\varepsilon_1^{\text{in}} = 2\ln(2)$. Nonetheless, this increase remains comparatively small when we consider large numbers of orders.

6 Future work

In this work, we have initiated the study of differential privacy in the trusted curator model, resulting in a definitional framework of round-differential-privacy, which protects both private inputs and private, yet correlated-outputs. We argue this setting applies to many economic or financial application domains. We introduce round-differentially-private market mechanisms for traditional finance, but also decentralized finance when instantiated

with privacy-preserving smart contracts [4].

We highlight the investigation of *general* correlated-output-differentially-private mechanisms for common output correlation classes as an interesting avenue for future work. In the setting of standard differential privacy, the Laplace, Gaussian or exponential mechanisms provide "plug-and-play" techniques to transform query algorithms into differentially privacy mechanisms. The investigation of similarly general techniques to achieve correlated-output-differential-privacy would represent a useful toolkit in the trusted curator setting. Achieving efficiency for such generalized mechanisms with custom MPC protocols would greatly facilitate the deployment of round-differentially-private mechanisms in practice.

References

1 Abbas Acar, Z Berkay Celik, Hidayet Aksu, A Selcuk Uluagac, and Patrick McDaniel. Achieving secure and differentially private computations in multiparty settings. In *2017 IEEE Symposium on Privacy-Aware Computing (PAC)*, pages 49–59. IEEE, 2017. https://doi.org/10.1109/PAC.2017.12.

2 Mehrdad Aliasgari, Marina Blanton, Yihua Zhang, and Aaron Steele. Secure computation on floating point numbers. *20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA*, 2013.

3 Abdelrahaman Aly, Kelong Cong, Daniele Cozzo, Marcel Keller, Emmanuela Orsini, Dragos Rotaru, Oliver Scherer, Peter Scholl, Nigel P. Smart, Titouan Tanguy, and Tim Wood. SCALE-MAMBA v1.12: Documentation, 2021. URL: https://homes.esat.kuleuven.be/~nsmart/SCALE/Documentation.pdf.

4 Carsten Baum, James Hsin-yu Chiang, Bernardo David, and Tore Kasper Frederiksen. Eagle: Efficient Privacy Preserving Smart Contracts. *Cryptology ePrint Archive*, 2022. https://eprint.iacr.org/2022/1435.

5 Jonas Böhler and Florian Kerschbaum. Secure multi-party computation of differentially private median. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2147–2164. USENIX Association, August 2020. URL: https://www.usenix.org/conference/usenixsecurity20/presentation/boehler.

6 John Cartlidge, Nigel P Smart, and Younes Talibi Alaoui. MPC joins the dark side. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, pages 148–159, 2019. https://doi.org/10.1145/3321705.3329809.

7 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. https://doi.org/10.1002/isaf.1502.

8 David Chaum, Debajyoti Das, Farid Javani, Aniket Kate, Anna Krasnova, Joeri De Ruiter, and Alan T Sherman. cmix: Mixing with minimal real-time asymmetric cryptographic operations. In *Applied Cryptography and Network Security: 15th International Conference, ACNS 2017, Kanazawa, Japan, July 10-12, 2017, Proceedings 15*, pages 557–578. Springer, 2017. https://doi.org/10.1007/978-3-319-61204-1_28.

9 David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981. https://www.doi.org/10.1145/358549.358563.

10 Tarun Chitra, Guillermo Angeris, and Alex Evans. Differential privacy in constant function market makers. *Cryptology ePrint Archive*, 2021. https://eprint.iacr.org/2021/1101.

11 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*, 2021. To appear at FC'22. https://eprint.iacr.org/2021/1549.

12 Mariana Botelho da Gama, John Cartlidge, Nigel P. Smart, and Younes Talibi Alaoui. All for one and one for all: Fully decentralised privacy-preserving dark pool trading using multi-party computation. Cryptology ePrint Archive, Paper 2022/923, 2022. https://eprint.iacr.org/2022/923. URL: https://eprint.iacr.org/2022/923.

13 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006. https://doi.org/10.1007/11681878_14.

14 Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014. http://dx.doi.org/10.1561/0400000042.

15 Fabienne Eigner, Aniket Kate, Matteo Maffei, Francesca Pampaloni, and Ivan Pryvalov. Differentially private data aggregation with optimal utility. ACSAC '14, page 316–325, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2664243.2664263.

16 Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1376–1385. JMLR.org, 2015.

17 Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 94–103. IEEE, 2007. https://doi.org/10.1109/FOCS.2007.66.

18 United States of America before the Securities and Exchange Commission. In the matter of itg inc. and alternet securities, inc., exchange act release no. 75672. https://www.sec.gov/litigation/admin/2015/33-9887.pdf, 12 Aug 2015.

19 United States of America before the Securities and Exchange Commission. In the matter of pipeline trading systems llc, et al., exchange act release no. 65609. https://www.sec.gov/litigation/admin/2011/33-9271.pdf, 24 Oct 2011.

20 United States of America before the Securities and Exchange Commission. In the matter of liquidnet, inc., exchange act release no. 72339. https://www.sec.gov/litigation/admin/2014/33-9596.pdf, 6 Jun 2014.

21 Manas Pathak, Shantanu Rane, and Bhiksha Raj. Multiparty differential privacy via aggregation of locally trained classifiers. *Advances in neural information processing systems*, 23, 2010. https://proceedings.neurips.cc/paper_files/paper/2010/file/0d0fd7c6e093f7b804fa0150b875b868-Paper.pdf.

22 Sikha Pentyala, Davis Railsback, Ricardo Maia, Rafael Dowsley, David Melanson, Anderson Nascimento, and Martine De Cock. Training differentially private models with secure multiparty computation. *arXiv preprint arXiv:2202.02625*, 2022. https://arxiv.org/abs/2202.02625.

23 Penumbra. ZSwap documentation. https://protocol.penumbra.zone/main/zswap.html, 2023.

24 Monica Petrescu and Michael Wedow. Dark pools in european equity markets: emergence, competition and implications. *ECB Occasional Paper*, (193), 2017. https://doi.org/10.2866/555710.

25 Sameer Wagh, Xi He, Ashwin Machanavajjhala, and Prateek Mittal. Dp-cryptography: marrying differential privacy and cryptography in emerging applications. *Communications of the ACM*, 64(2):84–93, 2021. https://doi.org/10.1145/3418290.

26 Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965. https://doi.org/10.1080/01621459.1965.10480775.

A Proofs

▶ **Theorem 11.** *rDP-Volume-matching is* $(\varepsilon^{in} + \varepsilon^{out}, \delta^{out})$-$(\varepsilon^{out}, \delta^{out})$-*m-round-differentially-private.*

Proof. (Theorem 11) Follows directly from Lemma 13, and Lemma 14, stated and proven below, for the duration of the privacy epoch.

We note that round-differential-privacy must be analyzed in the presence of corrupted traders and a corrupted liquidity provider; even if the liquidity provider only interacts with the "outputs" of clients by compensating for liquidity mismatch, it can also potentially infer knowledge about honest inputs from its output, complicating the proof.

Thus, our proof strategy is summarized as follows; we first state and prove Lemma 12 to demonstrate input-differential-privacy against corrupted traders; output-differential-privacy against corrupted traders holds trivially, as trader outputs are independently sampled. Then we first demonstrate output-differential-privacy against corrupted liquidity providers in Lemma 13 (in isolation); finally, input-differential-privacy against both corrupted traders and corrupted liquidity provider is considered in Lemma 14, which leverages privacy bounds from both Lemmas 12 and 13. ◀

▶ **Lemma 12.** *rDP-Volume-matching is* $(\varepsilon^{in}, 0)$*-input differentially private against an adversary that sees the adversarial trade outputs.*

Proof. (Lemma 12) Let $\mathbf{x}, \mathbf{x}'$ be neighboring vectors of trade orders (Definition 2) submitted to the DP-volume matching algorithm. Suppose for the rest of the proof that the honest user submits a buy order in one of $\mathbf{x}, \mathbf{x}'$ and a dummy order in the other. The proof is the same in the case of a sell order. The key to the proof is that at most one adversarial order is affected in the deterministic matching by changing the honest user's trade order.

Let b^A and s^A denote the number of adversarial buy and sell orders, respectively. If $b^A < s^A$ the number of matches increases by 1 by changing the honest user's trade from a dummy to a buy order. Here, the $(b^A + 1)$'th adversarial sell order changes from unmatched to matched and all other orders are unaffected. For $b^A \geq s^A$ the number of matches is the same for both inputs. If the honest user's buy order is not matched changing the input has no impact on any matches. However, if it is matched the s^A'th adversarial buy order changes from matched to unmatched. All other adversarial trades are unaffected.

We need to show that the probability of the adversaries observing any trade output changing by at most a factor of at most $e^{\varepsilon^{in}}$ between the two inputs. It is easy to see that this holds for the case where no adversarial trades are changed in step [1a] so it remains to show this for the case when one trade is changed. Let j be the index of the trader whose trade was changed between match and unmatched in the deterministic phase and let t denote a binary vector of trade outcomes. From the independence of the samples in step [1b] and equations 1 & 2 we have for any t:

$$\prod_{i \in \mathcal{A}} \frac{\Pr[\,\mathsf{trade}_i = t_i \mid \mathbf{x}\,]}{\Pr[\,\mathsf{trade}_i = t_i \mid \mathbf{x}'\,]} = \frac{\Pr[\,\mathsf{trade}_j = t_j \mid \mathbf{x}\,]}{\Pr[\,\mathsf{trade}_j = t_j \mid \mathbf{x}'\,]} \leq \frac{e^{\varepsilon^{\mathsf{in}}} \cdot (1 + e^{\varepsilon^{\mathsf{in}}})}{(1 + e^{\varepsilon^{\mathsf{in}}})} = e^{\varepsilon^{\mathsf{in}}}$$

◀

▶ **Lemma 13.** *rDP-Volume-matching is* $(\varepsilon^{out}, \delta^{out})$*-correlated-output-differentially-private.*

Proof. (Lemma 13) As per Definition 5, we must demonstrate that adversary output event probability distributions are $(\varepsilon^{\mathsf{out}}, \delta^{\mathsf{out}})$-indistinguishable to a change in the honest user's

output; the adversarial output view is composed of corrupted trader outputs and the view of the corrupted liquidity provider.

Note that inputs are fixed in correlated-output-differential-privacy. Thus, the output $\mathsf{match} = [\mathsf{match}_1, ..., \mathsf{match}_n]$ of deterministic matching in step [1a] remains unaffected; the distribution of trader outputs also remain unchanged in step [1b]. Correlated-output-differential-privacy holds trivially for the adversarial *trader output view*.

The *corrupted liquidity provider* provides reserves $(x_0^{\mathsf{liq}}, x_1^{\mathsf{liq}})$ and observes the updated reserves $(y_0^{\mathsf{liq}}, y_1^{\mathsf{liq}}) = \left(x_0^{\mathsf{liq}} + \Delta_0 - \rho_0\ ,\ x_1^{\mathsf{liq}} + \Delta_1 - \rho_1\right)$, where Δ_0 and $\Delta_1 := -\Delta_0$ are based on the liquidity mismatch from step [1b] and ρ_0 and ρ_1 are noisy values as described in step [2b]. The sensitivity of Δ_0 to the honest output is 1. An adversary who knows the adversarial trade outputs can compute the mismatch between them and therefore knows Δ_0 to an error of at most 1. For simplicity of presentation we assume that there is no mismatch between adversarial sell and buy orders. For the rest of the proof we assume that the honest user issued a buy order and S^h is the event where the order was not fulfilled. That is, $\Delta_0 = 0$ and $\Delta_0' = 1$. The other cases follow from symmetric proofs.

We split the outputs into two categories. For any event where $y_0^{\mathsf{liq}} < x_0^{\mathsf{liq}}$ we know that $\rho_0 > 0$ liquidity was frozen when conditioning on S^h. At the same time $\rho_0 - 1$ liquidity was frozen when conditioning on S^h not happening. We can see directly from the probability mass in Equation (4) that the conditional probabilities of observing any such output differ by a factor $e^{\varepsilon^{\mathsf{out}}}$. The probability of observing the special case of $y_0^{\mathsf{liq}} = x_0^{\mathsf{liq}}$ is 0 when the buy order was fulfilled because it implies that $\rho_0 = -1$. In contrast the probability is δ^{out} when conditioning on S^h. Therefore the algorithm satisfies $(\varepsilon^{\mathsf{out}}, \delta^{\mathsf{out}})$-correlated-output-differential-privacy, since for any event S^A we have

$$\Pr[\ \mathbf{M}^{\mathcal{A}}(\mathbf{x}) \in \mathcal{S}^{\mathcal{A}} \mid \mathbf{M}^{h}(\mathbf{x}) \in \mathcal{S}^{h}\ ] \leq \exp(\varepsilon) \cdot \Pr[\ \mathbf{M}^{\mathcal{A}}(\mathbf{x}) \in \mathcal{S}^{\mathcal{A}} \mid \mathbf{M}^{h}(\mathbf{x}) \notin \mathcal{S}^{h}\ ] + \delta.$$

◀

▶ **Lemma 14.** *rDP-Volume-matching is $(\varepsilon^{in} + \varepsilon^{out}, \delta^{out})$-input-differentially-private.*

Proof. (Lemma 14) We can consider rDP-Volume-matching as consisting of two separate algorithms. The first algorithm runs only the fuzzy order matching and reveals the trade outcomes to each trader. We know from Lemma 12 that this algorithm is $(\varepsilon^{\mathsf{in}}, 0)$-DP. The second algorithm takes the trade outcomes as input and runs the liquidity compensation step of the rDP-Volume-matching. The same proof as in Lemma 13 shows that this algorithm $(\varepsilon^{\mathsf{out}}, \delta^{\mathsf{out}})$-DP if we change honest input between a dummy order and a (possible fulfilled) valid order. By composition running the two algorithms satisfies $(\varepsilon^{\mathsf{in}} + \varepsilon^{\mathsf{out}}, \delta^{\mathsf{out}})$-input differentially privacy and the output distribution is equivalent to rDP-Volume-matching. ◀

▶ **Theorem 15.** *rDP-double auction is m-round-differentially-private.*

Proof. (Theorem 15) DP-double auction receives private trade orders and subsequently releases (1) a public clearing price $r \in \mathbf{r}$ from an $(\varepsilon_1^{\mathsf{in}})$-input-differentially private exponential mechanism, and (2) private trade outputs to each of the n clients from a $(\varepsilon_2^{\mathsf{in}}, \delta^{\mathsf{in}})$-$(\varepsilon^{\mathsf{out}}, \delta^{\mathsf{out}})$-round-differentially private DP-volume-match mechanism.

To establish round-differential privacy, we must argue correlated-output-differential-privacy for (1). This holds trivially, as trader outputs are determined by the DP-volume matching subroutine, where all trade outcomes are sampled independently from the clearing price. Thus, DP-double auction is $(\varepsilon_1^{\mathsf{in}} + \varepsilon_2^{\mathsf{in}}, \delta^{\mathsf{in}})$-$(\varepsilon^{\mathsf{out}}, \delta^{\mathsf{out}})$-round-differentially-private. m-round-differential-privacy is implied as long as liquidity frozen in each rDP-volume-match execution remains secretly locked. ◀

B Experimental Results

Here we provide the precise numerical values for the results presented in Section 5.3.

Table 1 Runtimes in seconds for the online phase of the rDP-volume-match algorithm.

Orders	InputCheckVM	MatchVol	Total
10	0.001	0.003	0.004
50	0.007	0.010	0.016
100	0.013	0.030	0.033
500	0.065	0.092	0.157
1000	0.130	0.184	0.314
5000	0.650	1.137	1.787
10000	1.300	2.638	3.938

Table 2 Runtimes in seconds for the online phase of the rDP-double-auction algorithm with $\varepsilon_1^{\text{in}} = 2\ln(2)$. Note that the FindPrice procedure includes both price determination and order matching.

Price points	Orders	InputCheckDA	FindPrice + MatchVol	Total
10	10	0.003	0.043	0.046
	50	0.015	0.051	0.066
	100	0.030	0.062	0.092
	500	0.148	0.137	0.285
	1000	0.296	0.232	0.528
	5000	1.479	1.222	2.701
	10000	2.957	2.757	5.714
100	10	0.015	0.391	0.405
	50	0.073	0.390	0.462
	100	0.145	0.404	0.550
	500	0.727	0.509	1.236
	1000	1.454	0.622	2.076
	5000	7.270	1.886	9.156
	10000	14.541	3.783	18.324

Table 3 Runtimes in seconds for the online phase of the rDP-double-auction algorithm with $\varepsilon_1^{\text{in}} = \ln(2)$. Note that the FindPrice procedure includes both price determination and order matching.

Price points	Orders	InputCheckDA	FindPrice + MatchVol	Total
10	10	0.003	0.149	0.152
	50	0.015	0.157	0.172
	100	0.030	0.168	0.198
	500	0.148	0.243	0.391
	1000	0.296	0.338	0.634
	5000	1.479	1.328	2.807
	10000	2.957	2.862	5.820
100	10	0.015	1.359	1.373
	50	0.073	1.358	1.430
	100	0.145	1.372	1.518
	500	0.727	1.447	2.204
	1000	1.454	1.590	3.044
	5000	7.270	2.854	10.124
	10000	14.541	4.751	19.292

Table 4 Runtimes in seconds for the online phase of the rDP-double-auction algorithm with $\varepsilon_1^{\text{in}} = \ln(2)/2$. Note that the FindPrice procedure includes both price determination and order matching.

Price points	Orders	InputCheckDA	FindPrice + MatchVol	Total
10	10	0.003	0.279	0.282
	50	0.015	0.286	0.301
	100	0.030	0.298	0.327
	500	0.148	0.373	0.521
	1000	0.296	0.468	0.764
	5000	1.479	1.457	2.936
	10000	2.957	2.992	5.949
100	10	0.015	2.566	2.580
	50	0.073	2.565	2.638
	100	0.145	2.580	2.725
	500	0.727	2.684	3.411
	1000	1.454	2.797	4.251
	5000	7.270	4.061	11.331
	10000	14.541	5.958	20.499