Privacy-Preserving Decentralized Exchange Marketplaces

Kavya Govindarajan
IBM Research India
kavya.g@ibm.com

Dhinakaran Vinayagamurthy
IBM Research India
dvinaya1@in.ibm.com

Praveen Jayachandran
IBM Research India
praveen.j@in.ibm.com

Chester Rebeiro
IIT Madras, India
chester@cse.iitm.ac.in

***Abstract*—Decentralized exchange markets leveraging blockchain have been proposed recently to provide open and equal access to traders, improve transparency and reduce systemic risk of centralized exchanges. However, they compromise on the privacy of traders with respect to their asset ownership, account balance, order details and their identity. In this paper, we present Rialto, a fully decentralized privacy-preserving exchange marketplace with support for matching trade orders, on-chain settlement and market price discovery. Rialto provides confidentiality of order rates and account balances and unlinkability between traders and their trade orders, while retaining the desirable properties of a traditional marketplace like front-running resilience and market fairness. We define formal security notions and present a security analysis of the marketplace. We perform a detailed evaluation of our solution, demonstrate that it scales well and is suitable for a large class of goods and financial instruments traded in modern exchange markets.**

***Index Terms*—Decentralized Exchange, Multi-Party Computation**

I. Introduction

Exchange markets have grown to be the cornerstone of trade and a measure of the health of an economy. While modern marketplaces provide sophisticated services including regulatory oversight, historically they provided three critical functions. First, they matched buyers and sellers, previously unknown to each other, to enable trade. Second, they facilitated a fair exchange between a buyer and seller by decoupling the exchange between them into two contracts, such that they did not have to enter into a contract with each other, but instead each have a contract with the exchange which was much more reputed and one with significantly lower counterparty risk of settlement. Third, the market permitted price discovery, enabling traders to learn the current fair price determined by market economics, at which they could buy or sell goods. In lieu of these services, the exchange would charge a transactional fee for each trade.

Over time exchange markets have grown significantly in size and volume and command high profit margins. Complex cumbersome processes that traders must follow have created an ecosystem of intermediaries who reduce transparency, increase costs and create higher barriers for entry. The global commodities trading forum report 2015 [28] cites transparency as one of the pressing concerns in the trading ecosystem. Further, centralized exchanges have also been the target of security attacks. A 2013 study [46] reported that 53% of exchanges surveyed had been hit by a cyber attack in 2012. A majority of these attacks were malware, denial of service and information theft attacks, and no financial theft was reported. Several centralized cryptocurrency exchanges such as Mt. Gox [36] have been hacked in recent years involving theft of several millions of dollars in cryptocurrency.

In an effort to provide equal access to all traders, improve transparency, and reduce systemic risk associated with the reliance on a trusted central entity for exchange operations, decentralized exchanges have been proposed in the recent past with regards to cryptocurrencies [1], [4], [9]–[11], [30] and commodities [3]. These decentralized exchanges leverage the cryptographic security and immutability of blockchain to track asset ownership, orders placed by traders and execute settlement in a distributed and replicated ledger. While they address many of the above issues with centralized exchanges, they raise two important challenges which have caused them to not be practical in reality.

First, centralized exchanges implicitly provide confidentiality of information pertaining to traders' asset ownership, account balances, orders prices and identity, with respect to other traders. The exchange itself acts as the sole and trusted custodian of this information, while only revealing information of recent trades in aggregate that is necessary for price discovery by the traders, for example, the top-K matched orders for some K. Current decentralized exchanges maintain this information in the clear on their ledgers, making it known to a larger set of entities that have a copy of the ledger. Prior research [21], [25], [26] has highlighted how public information regarding trades on blockchain have been leveraged in the past to launch front-running attacks. Further, financial privacy of asset ownership and trades is a key requirement in use cases such as private equity trading markets [39]. Brokers who assist in private equity trades are expected to maintain privacy of all financial transactions of their clients and only the centralized exchange is privy to the specific trades and their owners. An exchange also needs to be fair [40], where less competitive orders cannot be matched, while more competitive orders are unmatched.

Privacy-preserving payment networks on blockchain such as [7], [14], [18], [38] have been proposed, supporting confidentiality of transaction values along with user anonymity. In payment networks, it is inherently assumed that the trading parties know each other and have *agreed* for the transfer of certain assets or cryptocurrency. Zero knowledge proofs are constructed as per this prior agreement to ensure the integrity of the transfer while providing confidentiality of the assets and unlinkability of the trades to the buyer and the seller. In exchange markets, no such prior agreement exists. In the buy and sell orders we consider, no particular recipient is specified and only a constraint on the price is indicated. Moreover, to the best of our knowledge, while current state-of-the-art on decentralized privacy-preserving marketplaces provide front-running resilience which preserves order confidentiality until matching, they do not support order rate confidentiality throughout. How the matching problem based on these privacy constraints on orders and the settlement problem based on private account balances and orders are solved in a decentralized manner while providing unlinkability between the trades and the traders form the crux of this paper. Our fully decentralized solution without reliance on a trusted custodian or operator provides matching fairness and resilience against front-running and denial of service attacks. It also provides order rate confidentiality even after completion, revealing only minimal information necessary to support price discovery.

A second significant challenge of decentralized marketplaces today is scalability and performance. We demonstrate that our privacy-preserving decentralized marketplace is practical and scales to more than a thousand orders per minute, which is more than the trading volume for several classes of commodities in some of the largest centralized exchanges today. We envision that this performance will only increase with improvements in the underlying cryptographic building blocks. Additionally, a trader in our system only needs a one-time lightweight participation per trade.

Our main contributions in this paper are as follows:

- We propose Rialto, a fully decentralized privacy-preserving exchange market with matching, settlement and price discovery services, providing confidentiality to the trade rates and the account balances and unlinkability between the trades and the traders. Rialto also supports front-running resilience, market matching fairness and market availability, all while letting traders be arbitrarily malicious.
- We define formal security properties and present a security analysis of the marketplace, including a measure of privacy to quantify the information leakage (necessary to enable price discovery).
- We demonstrate through detailed experiments, that our proposed methods scale well and are suitable for a large class of commodities and financial instruments.

We describe the system model and problem statement in Section II. We discuss relevant background and provide an overview of our solution in Section III. We formally define the security properties of our system in Section IV. Details of the Rialto marketplace protocol are presented in Section V and in Section VI, we analyse the security of our protocol, quantify its privacy and discuss availability attacks. We discuss details of our evaluation in Section VII and conclude in Section VIII.

II. Problem Description

Our system model consists of a set of traders who wish to buy or sell an asset in a marketplace. The marketplace performs the function of matching buyers and sellers, and facilitates exchange and settlement. We assume that the traders place *limit orders*, which is a type of order to buy or sell the asset at a specified price or better. That is, *buy orders* specify the maximum price the buyer is willing to pay and *sell orders* specify the minimum price the seller is willing to sell at. For simplicity, we assume that each order is for exactly a unit quantity of the asset (any order for multiple units of the asset can be considered as multiple orders each of quantity one). Such an assumption does not support features such as discounts for bulk orders and does not optimize for cost of delivery. There could be an interesting future work for privacy-preserving matching with support for trade volume and privacy of asset quantity. We use the terms order price and rate interchangeably.

Note that an asset can be any physical, digital or virtual asset. and each order bid price may be specified in a standard currency that the marketplace operates in (this can be trivially extended to multiple types of assets that are traded in the marketplace and to even exchange between two assets directly). We only concern ourselves with recording the rightful ownership of the asset on a ledger managed by the marketplace after each trade and do not handle the physical exchange of assets.

All traders are registered and hold an account with the marketplace. We consider spot markets, where a trader must have sufficient balance in their account for posting a buy order and we do not consider margin buying (a scenario where the buyer purchases an asset using borrowed funds from a broker, with the purchased asset as collateral). A trader must hold the asset in order to post a sell order and we do not consider short selling (a scenario where a trader expects the price of the asset to go down, so sells the asset first and then buys it back within a specified time frame).

In many blockchain implementations including Bitcoin [37], the UTXO (Unspent Transaction Output) model has been popular rather than an account model. In the UTXO model, each transaction consumes a set of UTXOs and creates new UTXOs, possibly with a different set of owners. This makes checking for double spending easier as no two transactions can consume the same UTXO. However, in real-world scenarios and specifically for exchanges, the account model is a lot more practical and beneficial as additional functions such as KYC (Know Your Customer), authorization, deposit interest rates and loan rates can be supported. Hence, we use the account model in this work.

The marketplace records all buy and sell orders received in an order book. Periodically, it triggers a matching algorithm to match buyers to sellers such that their price constraints are satisfied, i.e., for each buyer and seller that are matched, the buyer's price is no less than the seller's price. For matched orders, the marketplace also determines a settlement price to complete settlement, wherein the buyer pays the seller the settlement price and the seller transfers ownership of the asset to the buyer. Orders not matched in one round are automatically carried forward to the next round of matching, up to a timeout after which they are expelled from the order book. This is necessary to ensure that the order book does not get clogged with irrelevant orders that are unlikely to be matched.

Traditionally, a centralized trusted organization with strict audit controls manages such a marketplace. Privacy is supported for the traders, but not with respect to the marketplace owner, who has access to all information. In this paper, we consider a decentralized marketplace of untrusted *facilitators* and endeavor to extend privacy to include the facilitators as well, and not just from other traders.

III. Solution Overview

A. Background

Pedersen Commitments: Commitment schemes are cryptographic techniques that allow one to commit to a value such that the commitment is both hiding and binding. The hiding property ensures confidentiality of the committed value, while the binding property ensures that the commitment cannot later be opened to another value. Pedersen commitment [41] (henceforth referred to as PC or just commitment) is a widely used commitment scheme which is also additively homomorphic.

Matching algorithm and Fairness: Many matching algorithms have been proposed in literature and are widely adopted by centralized exchanges today [29], [31]. Among them, the price-time algorithm is arguably the most popular. Buy orders are sorted in decreasing order of price and sell orders in increasing order, so that the most competitive orders by price are at the head of the lists. Orders are matched sequentially from the two lists until no more orders can be matched, that is the next buy order's price is less than the next sell order's price. We use a notion of fairness introduced in [40] in the context of double-sided auctions. A fair matching is fair to both buyers and sellers. A matching is fair to the buyers if there does not exist any unmatched buyer B_1, such that a buyer B_2 with order rate less than B_1 is matched. Similarly, the matching is fair to the sellers if there does not exist any unmatched seller S_1, such that seller S_2 with order rate more than S_1 is matched. The price-time algorithm is fair as it will always matches the most competitive orders first.

B. Decentralized Exchanges on Blockchain

The tamper-proof ledger of cryptographically signed transactions in a blockchain support the verifiability and accountability needed for decentralized exchanges. Smart contracts, which allow trusted distributed computation of functions, guarantee correctness and allow marketplace functionalities to be carried out in a trusted and risk-free manner. There are a large number of decentralized exchanges today for trading cryptocurrencies. Exchanges such as 0x Protocol [1] and IDEX [30] adopt a centralized off-chain matching engine with on-chain settlement. Airswap [4] and Bisq [10] expect traders to match each other (either off-chain or on-chain) by providing a peer-to-peer network for discovery and perform settlement on-chain and thus, do not offer automatic order matching. Binance DEX [9] and BlockDX [11] handle both matching and settlement on-chain. None of these exchanges support privacy of account balances or orders and store all information in the clear on the blockchain.

Other related work in the space of privacy-preserving decentralized exchanges include ZEXE, TEX, FuturesMEX, Dark MPC and P2DEX [6], [13], [17], [33], [35]. We provide a detailed comparison with our work in Table I with respect to the security properties we define in Section IV.

C. Design Alternatives

We provide an overview of various alternatives for designing a marketplace that exercise different trade-offs between performance and privacy. We categorize the alternatives across three dimensions of how information is shared, namely (i) level of decentralization of the marketplace, (ii) confidentiality of account balances, and (iii) confidentiality of order prices.

In recent years, decentralization of the marketplace has been proposed leveraging blockchain to reduce reliance on the trusted party, and improve security, transparency and operational costs by mitigating counterparty risk. However, an adverse side effect with decentralization is the additional loss of privacy. Traders' information, including their account balances and order details are now public to the decentralized set of entities operating the marketplace and anyone having access to the blockchain ledger. Prior work in supporting confidential payments on blockchain [38] has supported privacy of account balances leveraging ZKP.

As motivated earlier, it is not sufficient for only the account balances to be private. To prevent front-running and other attacks based on information theft, the orders placed by traders also needs to be protected. When account balances are public information, it does not help to keep the orders private as anyone who views the account balances before and after a transaction can discern the specific order price. So, we only consider protecting order information when the account balances are also private.

In order to match buyers and sellers, the marketplace needs to know if the price quoted by a seller is lesser than the maximum price a buyer is willing to pay. In other words, the marketplace needs to be able to perform comparison between buyer and seller prices to match orders. A central contribution of this paper is to support such a matching algorithm with encrypted prices and guarantee reliable and automated settlement in a decentralized manner.

One approach that supports partial privacy is inspired by the classic bucket sort algorithm [19]. The main idea of the bucketization based protocol is that the rate range is split into non-overlapping buckets, trading privacy for performance. Traders submit orders in two phases, wherein they submit their private rates as commitments in the first phase. The marketplace chooses random bucket start and end values after which traders disclose the corresponding public bucket values that their committed rate belongs to, in the second phase, along with a ZKP of range. This provides a coarse sorting of orders

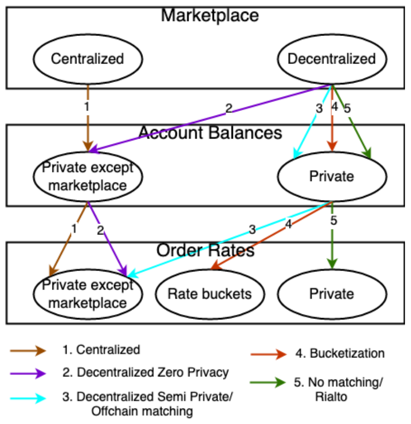

Fig. 1: Design alternatives for a marketplace

and the marketplace subsequently matches buyers in higher price buckets with sellers in lower price buckets. Matched orders are settled at the respective private rates of traders with additional zero knowledge proofs to guarantee integrity. We describe the order submission, matching and settlement of orders in this protocol in greater detail in Appendix A and use it as a baseline for comparison in our experiments in Section VII.

We propose Rialto, a privacy-preserving marketplace protocol. Secure multi-party computation (MPC) is one of the building blocks of our protocol. As MPC may not scale well to a large number of traders, and traders cannot be expected to always be online and reliably carry out computations needed by MPC algorithms, we introduce a set of *brokers*, fixed in number, to which the traders secret share the prices and enable them perform the necessary computations on private data. The MPC properties ensure that the individual brokers have no information regarding an order's price under reasonable collusion assumptions. Note here that a broker is a role performed by an entity. It is completely feasible that the facilitators operating the peers of the blockchain for the marketplace are themselves brokers, but for ease of exposition we describe them as separate entities from the blockchain peers. We refer to the brokers and the blockchain peers collectively as the marketplace.

Figure 1 pictorially represents the various alternatives mentioned above. The CENTRALIZED marketplace has a trusted third party operating the marketplace. Traders share their accounts and orders openly with the marketplace operator, who is trusted to maintain it privately, denoted by 'Private except marketplace'. The DECENTRALIZED ZERO PRIVACY mechanism replaces the trusted third party with a set of facilitators operating a blockchain, which is used to store account balances and orders. Here again, account balances and orders are private except to marketplace facilitators. The next alternative is DECENTRALIZED SEMI-PRIVATE, where account balances are maintained as commitments on the blockchain and not revealed to even the marketplace facilitators, but order details are shared with the marketplace in the clear. For both the above decentralized designs, it is possible to perform the matching component outside the blockchain by a trusted third party and only record the results on blockchain. This helps achieve greater performance at the cost of trusting the entity performing the off-chain matching. We consider this variant OFFCHAIN MATCHING as many decentralized exchanges today such as [1], [30] adopt this. Both the BUCKETIZATION scheme and Rialto support orders to also be private from the facilitators.

IV. SECURITY PROPERTIES

We now define the security properties of our system. Let κ be the security parameter. We have the marketplace operate in rounds with trades received up to a certain (possibly periodic) timer considered for matching in each round. Let N be the total number of traders in the marketplace. Let N_i be the total number of traders participating in round i and let T_i be the set of traders $\{t_{ij}\}_{j\in[N_i]}$ participating in round i. The view of a trader in the marketplace at the end of round i consists of the entire ledger till round i, its account balance at the beginning of each round $i' \in [i]$ and its order rate for each round $i' \in [i]$. We define the security properties from the perspective of a single malicious trader. These properties can be extended to capture a collusion among multiple malicious traders.

Property 1: *Confidentiality of Orders:* No information about the order rates except the top-K settlement rates is revealed at the end of a trading round.

Definition IV.1 (Order rate confidentiality)**.** There exists a probabilistic polynomial time (PPT) simulator Sim such that for every round i and for every adversarial trader t_{ij} in T_i, Sim simulates the view of t_{ij} when provided with (i) the top-K order rates for rounds $\leq i$, (ii) the account balance of t_{ij} at the beginning of round i and (iii) the order rate of t_{ij} for rounds $\leq i$. The probability for any PPT algorithm A to distinguish between the real view and the simulated view is $1/2 + \text{negl}(\kappa)$.

We use the notion of a *relaxed order rate confidentiality* for our system where the simulator is provided the *order book leakage*, which includes all the protocol specific leakages in addition to the top-K settlement rates.

Property 2: *Unlinkability of trades with traders:* An order cannot be linked to a trader who placed that order. Also, orders from different rounds cannot be linked back to the same (potentially unknown) trader who placed them. We formalize this property as follows.

Definition IV.2 (Trader unlinkability)**.** For any pair of rounds i and i', the probability of linking an order in round i placed by a trader in T_i to an order placed by the same trader in round i' is at most $1/min(N_i, N_{i'})$.

Property 3: *Confidentiality of Traders' Accounts:* No information is revealed about the account balance of a trader except the information inferred from the union of order book leakage across rounds.

Definition IV.3 (Account balance confidentiality)**.** We have an indistinguishability based security game between two PPT algorithms: a challenger C and an adversarial version A_t of trader t.

- A_t picks a trader t^*, and two values b_1 and b_2.
- C randomly chooses one among $\{b_1, b_2\}$ as the starting account balance for t^* in the first round it participates in. The marketplace proceeds with A_t orchestrating t and C orchestrating t^*.
- C aborts the game if the allowed order book leakage distinguishes between b_1 and b_2.
- A_t eventually outputs its guess b_1 or b_2.

The marketplace is said to have account balance confidentiality if the probability for A_t to correctly guess C's choice when C does not abort is $1/2 + \mathrm{negl}(\kappa)$.

Property 4: *Front-Running Resilience:* A malicious trader cannot learn information of an ongoing unmatched order to submit an order of their own and gain economic advantage.

Front-running resilience is ensured by the confidentiality of an order from other traders before the matching phase begins. Order rate confidentiality deals with hiding the order rates, even after matching and settlement are completed. While front-running resilience is supported by most marketplaces, order rate confidentiality is a more general privacy property to ensure only minimal necessary information about order rates is revealed, for price discovery.

Property 5: *Market Matching Fairness:* A less competitive buy or sell order (a buy order with a higher bid price or a sell order with a lower ask price) cannot be matched, when a more competitive order remains unmatched.

Property 6: *Market Integrity:* Once a buy and sell order are matched, the marketplace automates settlement. A trader attempting to deviate from this can be identified and penalized. Market integrity establishes the following:

- Buy rate is greater than or equal to the sell rate in each settlement. And there is no settlement which induces a buyer to pay more than its buy rate and a seller to be paid less than its sell rate.
- Each settlement ensures that the amount deducted from the buyer exactly equals the sum of amounts credited to the seller and to the marketplace as fees.

Property 7: *Market Availability and DoS Resilience:* The marketplace is designed to be resilient to Denial-of-Service and other attacks on its availability.

Our privacy-preserving decentralized marketplace enables automatic settlement and price discovery for traders in addition to the above security properties.

V. RIALTO: A PRIVACY PRESERVING DECENTRALIZED MARKETPLACE

An exchange marketplace has three main steps: order submission, matching and settlement. Rialto uses a blockchain ledger that maintains commitments of the account balances of all participating traders. At a high level, traders submit their orders by providing a commitment of their order rates to the ledger, and secret share their order rates to the brokers who perform a fair and maximal matching using MPC and a smart contract execution. We decide on an efficiency vs confidentiality tradeoff to move a part of the matching step outside MPC in a smart contract. Rialto supports price discovery which reveals the top-K order rates after settlement. So, we provide an analysis of this leakage in conjunction with that of the tradeoff. Settlement is done in a smart contract using the homomorphic property of the commitment scheme while retaining the confidentiality of order rates. Further, we make an efficient use of Waksman [47] network after settlement to perform an oblivious shuffle and achieve unlinkability of traders. During these steps, to protect against malicious brokers from modifying the input shares from the traders, we deploy an efficient distributed version of the Schnorr protocol [44] linking the shares to the corresponding commitments on the ledger. The components of Rialto are shown in Figure 2.

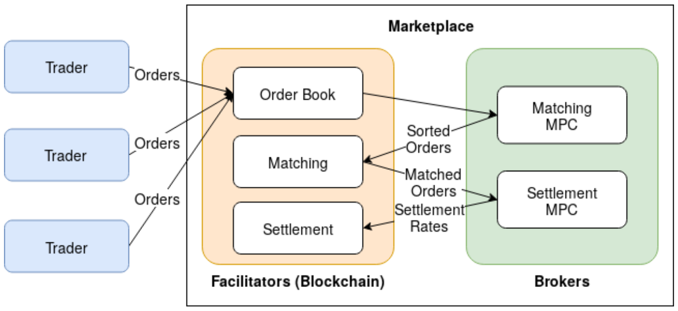

Fig. 2: Architecture diagram of the different components of the system for the Rialto protocol.

A. Trust model

We make the following trust assumptions on the entities in our system:

Traders: We assume all traders to be untrusted and financially incentivised to lie about asset ownership, account balances and attempt to cheat the system.

Blockchain facilitators: While individual facilitator nodes can be malicious, we assume a sufficient majority of them to be honest to preserve the integrity of the blockchain (e.g., two-thirds honest majority when using byzantine-tolerant consensus). The adversarial blockchain nodes may also collude with the traders.

Broker nodes: A broker node may collude with the traders, blockchain facilitators or other broker nodes. Depending on the requirements of the marketplace, adversarial broker nodes could be in minority or in majority. We present Rialto, a version of the protocol where the adversarial broker nodes are semi-honest (try to gain information without deviating from the protocol) and Rialto+ where they are malicious (can arbitrarily deviate from the protocol). Note that we treat the colluding traders and blockchain facilitators to be malicious in both these versions.

B. Protocol Description

Figure 3 provides a sequence of steps performed by the different entities involved in Rialto. Detailed algorithms are provided in Appendix E.

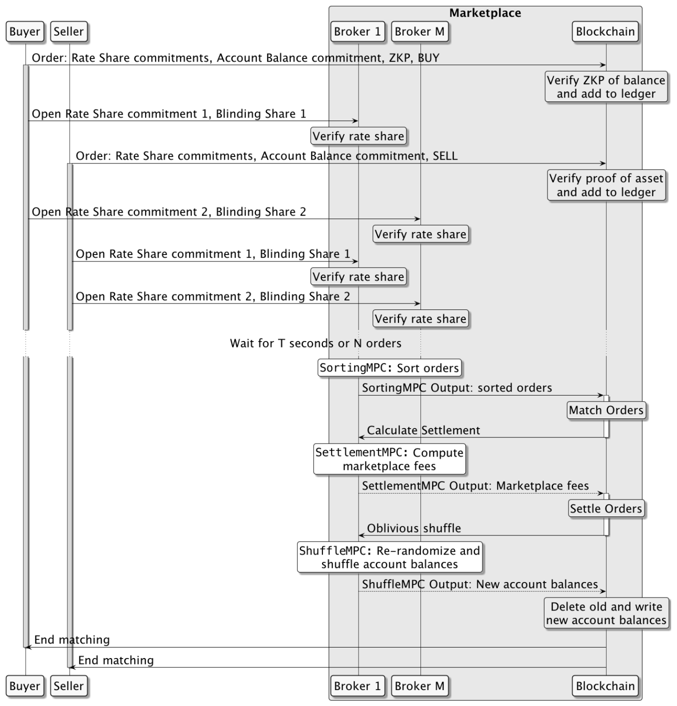
Marketplace
Buyer
Seller
Broker 1
Broker M
Blockchain
Order: Rate Share commitments, Account Balance commitment, ZKP, BUY
Verify ZKP of balance and add to ledger
Open Rate Share commitment 1, Blinding Share 1
Verify rate share
Order: Rate Share commitments, Account Balance commitment, SELL
Verify proof of asset and add to ledger
Open Rate Share commitment 2, Blinding Share 2
Verify rate share
Open Rate Share commitment 1, Blinding Share 1
Verify rate share
Open Rate Share commitment 2, Blinding Share 2
Verify rate share
Wait for T seconds or N orders
SortingMPC: Sort orders
SortingMPC Output: sorted orders
Match Orders
Calculate Settlement
SettlementMPC: Compute marketplace fees
SettlementMPC Output: Marketplace fees
Settle Orders
Oblivious shuffle
ShuffleMPC: Re-randomize and shuffle account balances
ShuffleMPC Output: New account balances
Delete old and write new account balances
End matching
End matching
Buyer
Seller
Broker 1
Broker M
Blockchain

Fig. 3: Sequence of calls for the Rialto protocol.

Order submission: Each trader secret shares its private rate value to get M additive shares, where M is the number of brokers in the system. The trader submits an order on the blockchain ledger containing commitments to each share. A buyer also submits a commitment of its account balance (as stored on the ledger) and a ZKP of range [15] that its order rate is less than or equal to its account balance. This ZKP is verified by a smart contract. Account balances are stored directly as commitments on the ledger, only the owner (which knows the value and blinding factor of the commitment) can submit a valid range proof. The trader also submits a share of the rate to each broker by opening the commitment of this share submitted on the blockchain. This secret sharing can be independent and outside of the MPC protocol used later by the brokers. Additionally, the trader submits a share of a random blinding factor to the brokers, which will be used to re-randomize the commitments of its account balance, to be described in Section V-C. In Rialto+, secret sharing scheme is decided based on trust expected from the broker nodes. For instance, we use a (k, n)-threshold secret sharing scheme [45], where $k = \lfloor n/2 \rfloor + 1$, to allow the protocol to proceed when a minority of the brokers are malicious and can abort. For a buy order, the blockchain locks funds equalling the buyer's rate in an escrow by using the commitment of the rate, computed by "adding" the commitments of its shares. The commitment of the buyer's account balance is also "subtracted" with the locked value. The marketplace allows only one trade order per account in a round, to prevent double spending.[1]

Sorting using MPC: The brokers wait for the earlier of T seconds or N orders and then perform MPC to determine the sorted list of orders. The SortingMPC algorithm takes N private inputs from each broker, where each input is a share of a trader's rate. The SortingMPC algorithm *securely* recomputes the shares and sorts the buy and sell orders together to output a single totally ordered list in an increasing order of prices. The information leaked as a result of this step is the sorted list of all buy and sell orders, revealing relative positions. The brokers finally submit the sorted list of orders to the blockchain smart contract, triggering the matching algorithm.

A malicious broker, colluding with a trader, can modify traders' rate share inputs, gaining it an advantage during matching. To prevent this, the brokers run the input share validation algorithm in Section V-D to validate the rate share inputs used in SortingMPC with the commitment of the rate shares which they obtain from the ledger. On completing the validation algorithm, the brokers continue with the SortingMPC algorithm using the validated shares.

Matching: After receiving the sorted list of orders from the brokers, a smart contract on the blockchain performs matching. The algorithm considers the buy orders and sell orders in ascending order. A buy order is matched with the first sell order whose rate is lower than the buyer's rate. This produces a maximal matching, the proof of which is presented in Appendix D. Trade orders that have not been matched in a round are carried forward to the next round. To prevent flooding of the marketplace with unmatched orders, we remove orders that have not been matched for more than a fixed number of rounds.

Settlement: For settlement, we use a scheme of type *Marketplace earns the rate difference*. This method of settlement is typically used as auctioneer's revenue in some double auctions [24], [48]. In this settlement scheme, each trader settles at their proposed rate and the difference between the buyer and seller rates is paid as fees to the marketplace. We settle between the matched buyers and sellers using the private proposed order rates as follows:

1) The marketplace invokes the brokers for computing the marketplace fees.
2) The brokers run the SettlementMPC algorithm with the rate shares of buyers and sellers as private inputs, and output a sum of all differences between buyer and seller rates. Note that the difference in rates between a specific matched buyer and seller is not revealed by this computation, but only the aggregate across all matched pairs of orders in a round is computed. If required, this can even released as an aggregate across multiple rounds.
3) The brokers submit the aggregate value to the marketplace, triggering the settlement algorithm on the blockchain.

[1]Multiple trade orders per account can be permitted, by the trader submitting the orders along with a ZKP that (i) sum of their orders is less than or equal to the account balance and (ii) each of their orders is greater than or equal to 0.

4) The smart contract settles matches by adding the commitment of seller's proposed rate to the seller's account balance, enabled by the homomorphism of the commitment scheme. It also removes the funds locked in the escrow on behalf of the buyer. The fees are added to the marketplace account.

The traders to be settled are a subset of the traders who submitted orders. For Rialto+, the input share validation performed before SortingMPC will ensure validation of inputs for SettlementMPC when continued with the validated shares.

After settlement, the difference between the old and new account balances of a trader equals the commitment of its order rate. This permits all parties to associate an account to its corresponding trade order, leading to linkability of accounts across rounds which affects account balance confidentiality and trader unlinkability. In section V-C, we propose a re-randomization and shuffling algorithm which occurs at the end of every round to un-link traders from accounts.

C. Re-randomization and shuffling

We propose a protocol using brokers to perform re-randomization and oblivious shuffle to achieve trader unlinkability. Re-randomization with an oblivious shuffle of the commitments prevents all parties, including brokers and blockchain facilitators, from associating accounts to trade orders by matching the difference of balance to order rates.

This is done as follows. Traders submit a secret share of a random blinding factor, which will be added to the randomness in the commitment of their account balance, for re-randomization. This blinding factor is submitted during order submission, so the protocol does not require participation from the traders during this phase. For oblivious shuffle, we use the Waksman permutation network [47]. When adhering to the setup in the rest of the protocol, we find this admits a better separation between the steps required to be performed inside the MPC and the steps brokers can perform outside and offline. Waksman network is a permutation circuit with N inputs and N outputs along with control bits which decide whether specific pairs of elements should be swapped. Other permutation networks such as Benes network have been shown to produce biased permutations [2] and will require multiple rounds to generate uniformly random permutations. Waksman network works by choosing a permutation uniformly and computing the corresponding control bits, which can be done offline by each broker prior to the MPC.

For each trader, each broker calculates the commitment of 0 using the blinding factor share offline. Brokers perform the following ShuffleMPC algorithm with the above commitments and control bits of their permutation as private inputs and trader account balance commitments as public inputs.

1) For each trader, brokers reconstruct the commitment of 0 with the blinding factor from submitted commitment shares and "add" it to the commitment of the trader's account balance.
2) Output of previous step is permuted by composing each broker's permutation network.

Finally, brokers output the re-randomized and shuffled account balance commitments to the marketplace. Marketplace removes the old account balance commitments stored in ledger and replaces with the new set. This does not affect marketplace integrity since the account balance values themselves do not change during the protocol. Also, a trader can compute the new commitment locally since the re-randomization is done using the randomness it submitted.

For N inputs and M parties, a single Waksman circuit is of depth $O(\log N)$ with $O(N \log N)$ swappers. On composing M networks, the total complexity of the shuffle is $O(MN \log N)$.

As before, malicious brokers can modify the blinding factor shares input to the protocol. This could prevent a trader from accessing its accounts in the future. For securing against malicious brokers, the integrity of the blinding factor shares used is validated using the input share validation algorithm before proceeding with the ShuffleMPC algorithm . As described in Section V-C, brokers also provide control bits for the Waksman network to permute the traders' account balances. A malicious broker may not input control bits for a uniformly random permutation network. However, honest brokers are guaranteed to perform a uniformly random permutation which when composed with other brokers' permutations will be uniformly random.

D. Input Share Validation Algorithm

Malicious brokers may not only deviate from the MPC protocol, but could also modify the inputs to the MPC algorithm. In Rialto, the private input shares to the MPC algorithms are grounded with their Pedersen commitments. The input share validation algorithm validates the input shares to the MPC by generating a ZKPoK of opening the commitments in a distributed manner. These proofs can be verified outside the MPC protocol avoiding exponentiation inside the MPC algorithm to validate the input shares.

Consider N secret values $v^t \; \forall t \in [N]$, whose commitments using blinding factor r^t are $c^t = g^{v^t} h^{r^t}$ and available to everyone. Each value v^t is secret shared along with its blinding factor r^t among M brokers, such that v_i^t and r_i^t are the shares for broker i and commitment $c_i^t = g^{v_i^t} h^{r_i^t}$ is public. Brokers would like to run an MPC algorithm $\mathcal{A}$ which takes as input the shares of secret values v^t from each trader.

We describe an input share validation algorithm $\mathcal{V}$ which will validate each broker's input according to its public commitment before executing $\mathcal{A}$. The proposed algorithm will involve an offline preparation phase, a validation phase involving an MPC and an offline verification phase.

Preparation phase: Each broker i generates random $y_i^t, s_i^t \leftarrow \mathbb{Z}_q$ and calculates $d_i^t = g^{y_i^t} h^{s_i^t}$ locally for each trader t.

Validation phase: The following steps are performed as MPC, taking y_i^t, s_i^t as private inputs and d_i^t as public input apart from $\mathcal{A}$'s private inputs of v_i^t and r_i^t. For each trader t,

1) For each share, calculate $e_i^t = Hash(g, h, y_i^t, s_i^t)$
2) For each share, calculate $a_i^t = y_i^t + e_i^t \cdot v_i^t$, $b_i^t = s_i^t + e_i^t \cdot r_i^t$
3) Output $\{a_i^t, b_i^t, d_i^t, e_i^t\}_{i \in [M]}$

Each broker then performs the following *offline verification phase*:

1) Consider an M bit vector $H = 1^M$.
2) For each trader t and for each broker i,

$$H[i] = H[i] \wedge \left(g^{a_i^t} h^{b_i^t} = d_i^t (c_i^t)^{e_i^t} \right)$$

On completing the verification, brokers continue with execution of algorithm $\mathcal{A}$ possibly with the brokers i such that $H[i] = 1$. Blockchain can be used to get a consensus on this set.

This distributed proof generation clearly doesn't affect the soundness of the generated proof since the view of the "verifier" subsumes the view of a verifier in the non-distributed version. Confidentiality of the distributed proof generation follows from the properties of the MPC protocol and the zero-knowledge property of the proof generation algorithm, along with the hiding property of the commitment scheme assumed for the non-distributed version.

E. Matching Fairness

Our proposed matching algorithm is fair to sellers, since sellers are considered in ascending order of rates. However, it is not fair to buyers, since buyers are also considered in ascending order of rates. This makes it possible that buyers with the most competitive price quotes are left unmatched.

We propose a swapping of matched buyers with any unmatched buyers to introduce fairness in our matching without affecting its maximality. This is similar to the swapping procedure in [40]. The algorithm swaps unmatched buyers with matched buyers lesser than it, by giving precedence to buyers with higher rates. Since after each swap the matches continue to remain feasible and there are at most n swaps, where n is the cardinality of the maximal matching, the final matching is both maximal and fair.

F. Price Discovery

Typically marketplaces reveal certain statistics regarding the settled orders such as top-K settlement rates at the end of each round, which helps in price discovery for buyers and sellers for subsequent rounds. K is usually a small percentage of N, the total number of orders in the round. Price discovery is important, since it allows future traders to determine their limiting rates. We explain how Rialto computes the top-K settlement rates at the end of each round. Our modified fair and maximum matching algorithm after swapping has the property that among matched buyers and sellers (B_1, S_1) and (B_2, S_2), if B_1 has a higher order rate than B_2 then S_1 is no less than S_2. This implies that the top-K settlement rates are for the matches of the top-K buyers, where the settlement rate is considered as the buyer's proposed rate for the scheme where marketplace earns the difference. Where order rates are shared publicly with the marketplace, the marketplace facilitators can easily compute the settlement rates of the K highest buyers and their matches and reveal it. In Rialto, we do it by letting the MPC reveal the buyer's rate value (by reconstruction of shares) for each of the top-K buy orders (after sorting) which have matched sell orders.

VI. Security Analysis

In this section, we analyse the security of our protocol. In Section VI-A, we discuss how confidentiality of traders accounts and order rate is maintained (Properties 1 to 4 in Section IV). We also discuss the integrity of Rialto marketplace (Property 6 in Section IV) in Section VI-B. In Section VI-C, we quantify order rate privacy based on the information leaked by the system to measure the privacy of the protocols against marketplace participants. In Section VI-D, we discuss availability and DoS resilience of Rialto. In Section VI-E, we compare our work with related work on privacy preserving decentralized exchanges, along the axis of our security properties.

A. Confidentiality and Unlinkability

We argue the confidentiality and unlinkability properties of our system at a high level in this section. As mentioned in Section IV, the properties are defined with respect to a single malicious trader which can collude with the adversarial blockchain and broker nodes and these can be extended to capture a collusion among multiple malicious traders.

The high-level arguments for the claims presented in this section also assume semi-honest broker nodes. We already expect the semi-honest broker nodes to collude with the malicious blockchain nodes and traders. The colluding traders can reveal their order rates and account balances to the broker nodes, while the blockchain nodes do not input secret information in any of the rounds. Hence, the extension of the arguments to malicious broker nodes involves proving that the broker nodes cannot use the colluding traders' order rates and account balances to deviate from the protocol and gain more information about honest traders than what's gained by the semi-honest broker nodes colluding with the malicious traders and blockchain nodes. An observation that will help here is that the inputs to the broker nodes are secret shares of all the private inputs from the traders and their commitments, and hence the collusion with a subset of traders reduces the MPC to that with inputs as the shares from the honest traders.

Order book leakage: The consolidated order book leakage in Rialto for round i is as follows: for each round $\bar{i} \leq i$, (i) top-K settlement rates for round $\bar{i}$, (ii) the ordering between the order rates associated with the trader indexes $j \in [N_{\bar{i}}]$, (iii) for each $j \in [N_{\bar{i}}]$, the next round that this account participates in, (iv) aggregate marketplace fees for that round.

Claim 1. Rialto provides relaxed order rate confidentiality.

The simulator is provided the order book leakage and it sets the order rates of each trader in a way that matches the order book leakage. We describe how the simulator outputs are generated from the order book leakage while being indistinguishable to their real world counterparts. During the order submission phase, information about an order rate is only used in the following ways: (i) commitment on the order rate (ii) the secret shares of the order rates to the broker nodes.

The hiding properties of the Pedersen commitments and the secret sharing protocol help in indistinguishability between the real and the simulated outputs.
After the order submission phase, information about order rates are used in the following entities that are revealed: (i) the ordering between the order rates, (ii) the top-K settlement rates for the round, (iii) the aggregate of marketplace fees for that round, and (iv) secret shares of the difference between the buyer and seller rates (as marketplace fees) to the broker nodes. The first three are part of order book leakage input to the simulator and the simulator chooses the order rates to match with these outputs. The fourth entity can be easily generated in accordance with the aggregate marketplace fees.

Claim 2. Rialto provides front-running resilience.

Rialto reveals no information on the order rates for the ongoing round before the matching phase begins, and this ensures front-running resilience. The argument for this property is similar to the one in Claim 1, where the simulator obtains the order book leakage for all the previous rounds (and not for the ongoing round). The order rates for the ongoing round are chosen arbitrarily to match the order book leakage from the previous rounds. The hiding properties of the Pedersen commitments and the secret sharing protocol help in indistinguishability between the real and the simulated outputs for the ongoing round. And this proves that there is no leakage on the order rate (before the matching phase begins) since the simulator input here does not contain the order rate for the final round.

Claim 3. Rialto provides account balance confidentiality.

We will argue that an adversary cannot differentiate between the two possible initial account balances if they admit the same order book leakage across rounds. The hiding property ensures that the commitment scheme does not reveal information about account balance when a trader enters the marketplace. This along with the security of the secret sharing protocol and MPC protocols ensure account balance confidentiality.
The functions that involve account balance of a trader are (i) the comparison with the buy rate (for the buyer accounts) during submission, (ii) subtraction/addition of buy/sell rates during settlement. Hence, the information revealed about account is limited to the union of order book leakage for the rounds that a polynomial time adversary can associate this account to[2].

Claim 4. Rialto provides trader unlinkability when account balance confidentiality and order rate confidentiality are provided.

We will argue that the probability for any adversary to link a pair of orders across rounds i and i' to a trader is the maximum among $1/N_i$ and $1/N_{i'}$. The order submission phase reveals the link between an account (i.e., commitment of account balance) that is participating in the round to the order it submits. But the oblivious shuffle protocol at the end of each round unlinks the account (old commitment) from its updated account (new commitment). Hence, the only information that is revealed after a round for all matched orders[3] is the subsequent round that each account (new commitment) in this round participates in. Any adversary is limited to a guess among the traders participating in the rounds, in addition to the negligible advantage in breaking account balance and order rate confidentiality and thus attempting to link traders through account balances. Hence it is the maximum of $1/N_i$ and $1/N_{i'}$ for rounds i and i'.

[2]The rounds of interest for an adversary to associate orders to this account is limited in a real world marketplace since the number of possibilities to keep track grows exponentially with the depth of the transaction graph rooted at the round where this account enters the marketplace, and this exponent has a large base with a large numbers of traders participating in each round.

B. Market Integrity

We state the integrity of the Rialto marketplace.

Claim 5. Rialto ensures market integrity.

At a high level, an adversary breaking market integrity with a non-negligible advantage breaks one of the following with a non-negligible advantage:

- soundness of the zero-knowledge proofs (order rate for a buyer is lesser than or equal to its account balance)
- binding property of the commitment scheme (order rate used to generate the proof is the same as what is committed to)
- correctness of the inputs used in MPC (semi-honest brokers are trusted to validate the input shares through their commitments on the blockchain; input share validation ensures this for malicious brokers)
- correctness of the MPC protocol among the brokers
- correctness of the smart contract execution (verify proofs and matching and settlement of orders as per the protocol)

C. Quantifying Order Rate Privacy

In Section VI-A, we prove that the leakage of order rates is atmost *order book leakage*.

While an adversary (a trader or broker) cannot uncover any individual order placed by a specific trader, they can endeavor to estimate the distribution of orders in a given round based on the information leaked post-settlement. We quantify the order book leakage to measure the privacy of the order rates. Specifically, we calculate the relative entropy of the estimated distribution of the order rates from its true distribution, in a trading round, expressed as a percentage of the entropy of the true distribution. This measures how closely adversaries can guess the true distribution. In Appendix B-A, we detail the analysis to measure the privacy gain (Equation 1).

$$\text{Privacy Gain} = \frac{\mathtt{KL}(\mathtt{P_E}, \mathtt{P_T})}{\mathtt{H}(\mathtt{P_T})} \times 100 \qquad (1)$$

[3]Unmatched orders might continue to remain in the next round and hence the anonymity set is restricted to the traders whose orders get matched. Such a marketplace behaviour is not accounted for in this analysis.

D. Discussion on Availability

We assume till now that (i) the brokers are available and (ii) blockchain network is available. These ensure that the marketplace is available.

The rest of the section discusses this assumption on the availability of the blockchain network and the broker nodes.

Flooding attack by malicious traders: A malicious trader can launch a DoS attack by flooding the marketplace with a large number of orders that are unlikely to be matched. The marketplace (i) excludes repeat orders from the same account during order submission and (ii) locks funds for each order placed before matching. Further, the marketplace could charge a fee from traders for each submitted order, which can be refunded upon settlement. Unmatched orders are terminated after a predetermined number of rounds to refresh the order book.

Denial of service attacks on brokers: A targeted attack on a single broker might restrict its participation in the MPC after receiving rate shares. Note that this is covered with the attacked brokers being malicious.

Denial of service attacks on blockchain: The decentralized nature of blockchain is naturally resistant to DoS attacks. Having more blockchain peer nodes increases DoS resistance and is a well studied topic. In certain blockchain platforms, it may be possible for a malicious peer or miner to delay or deny a trader from placing their order on blockchain. This requires care in designing the blockchain platform to be resilient against malicious peers.

E. Comparison with Related Work

In Table I, we present a comparison of our work against relevant related work. ● /○ denote presence/absence of property in cited work. ◐ denotes the relaxed order rate confidentiality property of our work.

FuturesMEX presents a distributed futures exchange where the order book is publicly visible, but trader anonymity is ensured. It supports trader and inventory (account balance) anonymity, but does not provide order rate confidentiality or resilience to front-running attacks. We do not consider futures contracts where trade positions are open for longer than a round of matching and settlement, and their attack scenarios are not directly applicable to our system that deals only with spot trades. It would nevertheless be a very interesting future work to combine the fully private order book in our system with futures contracts and margin buying supported by FuturesMEX.

TEX is a centralized marketplace protocol which supports front-running resilience through its time lock orders and receipts. However, the centralized operator learns the order rate after the trader reveals the secret key for matching. Also, while the settlement is non-custodial and decentralized, by verification using ZKPs, account balances are revealed to the operator.

ZEXE introduces a protocol for private, secure off-chain computation by an operator with public on-chain verifiable transactions. Their closed-book DEX supports order rate confidentiality, front-running resilience and trader anonymity against traders but not against the non-custodial order book operator. The settlement and account balances are private to traders but not to the order book operator. Since order rates are private to traders, they do not support price discovery.

Dark MPC presents a privacy-preserving matching protocol for CDA (Continuous Double Auction) in dark markets using MPC. However, they do not support ownership and settlement of assets (represented by '-' in Table I under account balance privacy and decentralized settlement). Also, their protocol does not support price discovery.

A concurrent work, P2DEX presents a privacy-preserving decentralized exchange using publicly verifiable MPC to perform privacy-preserving matching on orders. However, settlement transactions, using the UTXO model, are on plaintext order prices, thereby not supporting order rate confidentiality. Prior work [42], [43] has shown that transaction tracing and transaction graph analysis results in de-anonymization of Bitcoin, affecting both anonymity and account balance privacy.

In Appendix C, we discuss about Automated Market Maker and briefly describe the design of a privacy-preserving AMM DEX.

VII. Evaluation

In this section, we present the details of the implementation and the performance evaluation of the Rialto protocol.

A. Implementation

Blockchain Platform: We implemented the decentralized blockchain based marketplace protocols using Hyperledger Fabric [5]. It is a permissioned blockchain platform which, as opposed to permissionless blockchains, leverages strong identities based on digital certificates to enable permissioned membership to the blockchain for both peer nodes as well as clients. It has support for smart contracts with the ability to restrict access to functions and data to specified roles and identities. Traders are implemented as clients with unique identities on the blockchain. Marketplace functionalities of managing account balances for traders, maintaining an order book, and performing matching and settlement are implemented as smart contracts (*chaincodes*) on Hyperledger Fabric, written in Golang.

MPC: Brokers in Rialto are implemented using the open source MP-SPDZ framework [22], [32]. For the semi-honest protocol, we use the replicated-ring-party mode for 3 parties and shamir-party mode for more than 3 parties. For the malicious version, we use the ps-rep-ring-party mode.

Range proofs and Pedersen Commitments: All the proposed protocols require ZK proofs of range for proof of account balance. For the ZK proofs of range, we have used bulletproofs [15], which is a non-interactive ZK proof protocol which supports very efficient range proofs. We have used an open source Rust implementation of bulletproofs [20] which also provides support for Pedersen Commitments on the Edwards25519 elliptic curve. Since the proofs are verified by the marketplace facilitators on blockchain, we have written a Rust wrapper around the bulletproofs package which is

Related Work	Account Balance Confidentiality	Order Rate Confidentiality	Trader Anonymity	Front Running Resilience	Price Discovery	Decentralization	
						Matching	Settlement
FuturesMEX [35]	●	○	●	○	●	●	●
TEX [33]	○	○	●	●	●	○	●
ZEXE [13]	○	○	○	○	○	○	●
Dark MPC [17]	-	●	○	●	○	●	-
P2DEX [6]	○	○	○	●	●	●	●
Rialto (Our Work)	●	◐	●	●	●	●	●

TABLE I: Comparison with related work.

compiled as a static library and used with the smart contracts in golang using Cgo.

B. Evaluation

We present a detailed evaluation of the performance of the Rialto protocol and use the alternatives mentioned in Section III as baselines for comparison. Our goal is to demonstrate that we can achieve high levels of privacy and decentralization with acceptable trade-offs on performance. All experiments are performed on a 14-core 64-bit Intel Xeon Gold 6132 CPU @ 2.60GHz. The exchange smart contract is initialized with an account for each trader with a certain balance (or a commitment of the balance for our privacy-preserving schemes) and ownership of a certain quantity of the asset to be traded in the marketplace.

During each experiment, trader processes continuously submit buy and sell orders to the marketplace following a poisson process with a certain mean. Their order price is an integer drawn from a uniform distribution with a mean value of 255 for buyers and 245 for sellers and a variance of 15 (the distribution of prices does not impact performance). Periodically, matching is triggered on all outstanding trades submitted, followed by settlement. We refer to the completion of each instance of matching and settlement as a round. Each experiment lasts for 12 rounds and each data point in our plots is obtained by averaging across 12 experiments. Unless specified otherwise, we use the following default values: 512 orders submitted on average in each round, 3 brokers, round time of 30 seconds (that is, matching is triggered every 30 seconds) and a marketplace comprising of 4 blockchain peers. For the BUCKETIZATION scheme, we use a default bucket width of 4. We study the performance of our protocols by varying each of these parameters.

We measure the following metrics:

- *Average end-to-end latency:* This is the primary metric essential to a trader and is the average time taken from submission of an order until it is settled.
- *Component-wise latency:* The end-to-end latency comprises of waiting time until the next matching round, matching time, settlement time and for Rialto, the time taken for sorting using MPC.
- *Percentage of orders matched*
- *Marketplace fees* as a % of total worth of trades settled

Scaling number of orders: In this experiment, we compare the component-wise and end-to-end latency of all the proposed protocol variants for increasing number of orders in each round and present the results in Figure 4. Centralized commodity exchanges today handle up to a few hundred orders a minute per commodity. To demonstrate that our protocols can scale to such volumes, we present results for 128, 256, 512 and 1024 orders generated in each round comprising of 30 seconds. The non-privacy preserving, but decentralized protocol variants incur hardly any additional latency with increasing number of orders, showing that the use of blockchain smart contracts scales very well with load. The BUCKETIZATION and Rialto protocols exhibit a marginal sub-linear increase in latency with increasing load, significantly surpassing the scaling needs of most commodities traded in exchange markets today. For handling a large number of commodities in parallel, the underlying infrastructure can be suitably scaled.

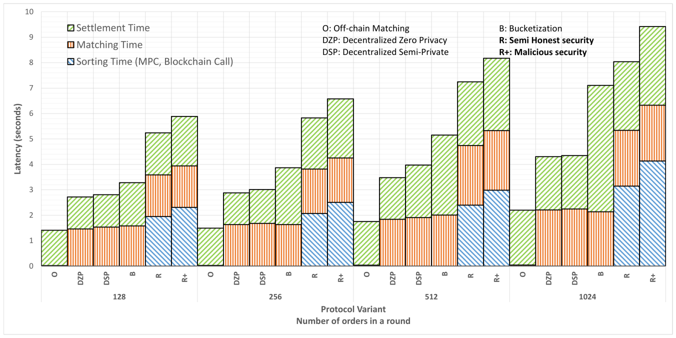

Fig. 4: Component-wise and end-to-end latency comparison of all 6 protocols for increasing order volume

The CENTRALIZED exchange (not shown in the graph) without any privacy or decentralization takes negligible time for matching and settlement and is the most performant. For the OFFCHAIN MATCHING protocol, matching is performed by a centralized entity outside blockchain and the main latency incurred is the cost of performing settlement on blockchain. The DECENTRALIZED ZERO PRIVACY protocol incurs a small added delay due to the use of blockchain for both matching and settlement. The DECENTRALIZED SEMI PRIVATE protocol, additionally leverages Pedersen commitments to store the account balances in private and hence incurs slightly higher settlement times. For the BUCKETIZATION protocol, since matching is based on public bucket floor and ceiling values, it is as performant as the previous protocols. However, settlement based on the private order rates of the traders incurs additional overhead for communication between the matched traders, recording transactions on blockchain and for zero knowledge proofs. As expected, this overhead during settlement is also higher for larger volume of orders, but scaled sub-linearly as

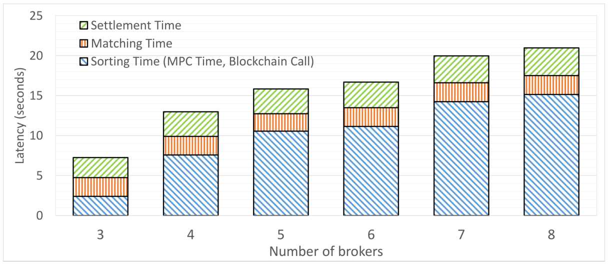

Fig. 5: Latency in Rialto for different number of brokers, for semi-honest security.

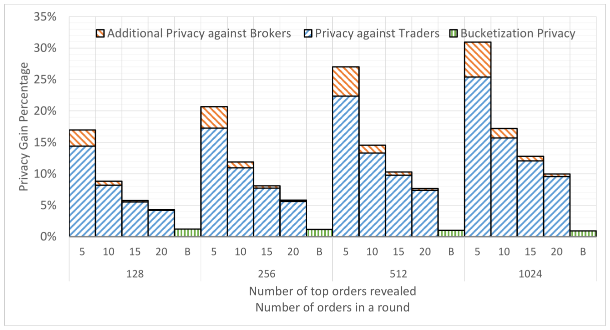

Fig. 6: Privacy of Rialto for traders and brokers for increasing order volume and number of top matched orders revealed.

matched pairs of buyers and sellers can execute the settlement process in parallel. Finally, the Rialto protocols experience the highest latencies for matching and settlement largely owing to the overhead of MPC. The SortingMPC time, although performed by a fixed set of 3 brokers, increases nearly linearly with the number of orders due to the complexity of sorting. With the duration of a round set to 30 seconds, all trade orders will have a waiting time (not shown in graph) of an average of 15 seconds for all the protocols. The marketplace is mostly idle during this time, except for verifying proofs.

Scaling number of brokers: The choice of number of brokers presents a crucial trade-off between performance and the level of tolerance to collusion among the brokers. Privacy is lost if all brokers collude to compute the secret order price of traders. Figure 5 presents a comparison of the component-wise latency for the Rialto protocol with semi-honest security when the number of brokers is varied from 3 to 8 with 512 orders submitted per round. With an increase in the number of brokers, the time taken by the MPC for sorting orders increases linearly, while matching and settlement times remain nearly the same.

Blockchain scalability: We varied the number of blockchain peer nodes from 4 to 12, which did not impact the latencies for the Rialto protocol (plot not shown). Our blockchain based marketplace scales quite well to a large number of blockchain nodes.

Communication Overheads: The communication costs in the system occur due to the blockchain and MPC components. Blockchain overheads are quite low with our use of Hyperledger Fabric, with blocks having a size of at most a few hundred kilobytes. For the default parameters of 512 orders in a round and 3 broker nodes, each party communicates 1.38 MB for MPC for semi-honest security and 20.79 MB for malicious security. This overhead is higher for a larger number of brokers, but is still easily achievable even across wide area networks.

Privacy Measure: As discussed in Section VI-C, we defined the privacy gain of the marketplace as the KL divergence of the estimated distribution from the true distribution as a percentage of the entropy of the true distribution.

For order rates sampled from a normal distribution and averaged across 100 runs, we computed the privacy gain of the Rialto and the BUCKETIZATION protocols for varying number of top matched orders revealed (K) and the number of orders (N) in each round. The results are presented in Figure 6. For the Rialto protocol, the marketplace has higher privacy against brokers than against traders, who have higher information gain from their own order rate. As expected, privacy gain is higher for lower K, as lesser information is revealed. For a fixed K, privacy gain increases with the number of orders in a round as many more values are hidden. The privacy of the BUCKETIZATION protocol does not vary much with K since adversaries can approximate other unknown order rates to the granularity of a bucket.

VIII. CONCLUSION

In this paper, we present the design of Rialto, a fully decentralized privacy-preserving exchange marketplace with matching, automated on-chain settlement and price discovery supporting marketplace properties such as order rate and account balance confidentiality, unlinkability between trade orders and traders, front-running resilience and fairness. We define formal security notions for the marketplace and present a security analysis of our protocol. We quantify information leakage to allow the marketplace to determine parameters in accordance with its privacy needs. We demonstrate that our proposed solutions scale well and are suitable for real world markets for a large class of commodities.

As described in our paper, marketplaces need to occasionally reveal the marketplace statistics to help future participants in discovering the current market price for an asset. However, this affects order rate confidentiality, presenting a trade-off between confidentiality and utility. One interesting future work would be to calculate and reveal such marketplace statistics in a differentially private manner without affecting the marketplace properties like market integrity.

REFERENCES

[1] 0x Protocol. https://0x.org.

[2] M. Abe and F. Hoshino. Remarks on mix-network based on permutation networks. In *International Workshop on Public Key Cryptography*, pages 317–324, 2001.

[3] Affogato Network. https://medium.com/affogato-network.

[4] Airswap. https://www.airswap.io/.

[5] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, and Y. Manevich. Hyperledger fabric: A distributed operating system for permissioned blockchains. In *ACM Eurosys*, 2018.

[6] C. Baum, B. David, and T. K. Frederiksen. P2dex: Privacy-preserving decentralized cryptocurrency exchange. In *Applied Cryptography and Network Security*, pages 163–194, 2021.

[7] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *IEEE SP*, 2014.

[8] H. Berg, T. A. Proebsting, et al. Hanson's automated market maker. *Journal of Prediction Markets*, 3(1):45–59, 2009.

[9] Binance DEX. https://www.binance.org/.

[10] Bisq. https://bisq.network.

[11] Blocknet. https://blocknet.co/block-dx/.

[12] G. Blom. *Statistical estimates and transformed beta-variables*. PhD thesis, Almqvist & Wiksell, 1958.

[13] S. Bowe, A. Chiesa, M. Green, I. Miers, P. Mishra, and H. Wu. Zexe: Enabling decentralized private computation. In *IEEE SP*, pages 820–837, 2018.

[14] B. Bünz, S. Agrawal, M. Zamani, and D. Boneh. Zether: Towards privacy in a smart contract world. In *FC*, pages 423–443, 2020.

[15] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *IEEE SP*, pages 315–334, 2018.

[16] A. Capponi and R. Jia. The adoption of blockchain-based decentralized exchanges: A market microstructure analysis of the automated market maker. *Available at SSRN 3805095*, 2021.

[17] J. Cartlidge, N. P. Smart, and Y. Talibi Alaoui. Mpc joins the dark side. In *ACM AsiaCCS*, pages 148–159, 2019.

[18] E. Cecchetti, F. Zhang, Y. Ji, A. Kosba, A. Juels, and E. Shi. Solidus: Confidential distributed ledger transactions via pvorm. In *ACM CCS*, 2017.

[19] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. Section 8.4: Bucket sort. *Introduction to Algorithms,*, pages 174–177, 2001.

[20] D. Cryptography. https://github.com/dalek-cryptography/bulletproofs.

[21] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels. Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges. 2019.

[22] C. Data61. Mp-spdz. https://github.com/data61/MP-SPDZ.

[23] B. David, P. Gaži, A. Kiayias, and A. Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Eurocrypt*, pages 66–98, 2018.

[24] K. Deshmukh, A. V. Goldberg, J. D. Hartline, and A. R. Karlin. Truthful and competitive double auctions. In *ESA*, pages 361–373, 2002.

[25] S. Eskandari, S. Moosavi, and J. Clark. Sok: Transparent dishonesty: front-running attacks on blockchain. In *FC*, 2019.

[26] S. Eskandari, S. Moosavi, and J. Clark. Transparent dishonesty: front-running attacks on blockchain. In *Financial Cryptography*, 2019.

[27] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *SOSP*, pages 51–68, 2017.

[28] Global Commodities Forum Report. Trade in commodities: Challenges and opportunities. https://unctad.org/en/PublicationsLibrary/suc2015d1_en.pdf.

[29] C. Group. Match algorithms. http://web.archive.org/web/20120626161034/http://www.cmegroup.com/confluence/display/EPICSANDBOX/Match+Algorithms.

[30] Idex. https://idex.market/eth/idex.

[31] K. Janeček and M. Kabrhel. Matching algorithms of international exchanges, 2007.

[32] M. Keller. Mp-spdz: A versatile framework for multi-party computation. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1575–1590, 2020.

[33] R. Khalil, A. Gervais, and G. Felley. Tex - a securely scalable trustless exchange, 2019. https://eprint.iacr.org/2019/265.

[34] D. Kraft. Game-theoretic randomness for blockchain games. *arXiv preprint arXiv:1901.06285*, 2019.

[35] F. Massacci, C. N. Ngo, J. Nie, D. Venturi, and J. Williams. Futuresmex: Secure, distributed futures market exchange. In *IEEE SP*, pages 335–353, 2018.

[36] Mt. Gox. https://en.wikipedia.org/wiki/Mt._Gox.

[37] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. http://www.bitcoin.org/bitcoin.pdf.

[38] N. Narula, W. Vasquez, and M. Virza. zkledger: Privacy-preserving auditing for distributed ledgers. In *Usenix NSDI*, 2018.

[39] Nasdaq Private Market. https://www.nasdaq.com/solutions/private-company-solutions.

[40] J. Niu and S. Parsons. Maximizing matching in double-sided auctions. In *AAMAS*, page 1283–1284, 2013.

[41] T. P. Pedersen. A threshold cryptosystem without a trusted party. In *Eurocrypt*, pages 522–526, 1991.

[42] F. Reid and M. Harrigan. An analysis of anonymity in the bitcoin system. In *Security and privacy in social networks*, pages 197–223. Springer, 2013.

[43] D. Ron and A. Shamir. Quantitative analysis of the full bitcoin transaction graph. In *FC*, pages 6–24, 2013.

[44] C.-P. Schnorr. Efficient identification and signatures for smart cards. In *Crypto*, pages 239–252, 1989.

[45] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[46] R. Tendulkar and G. Naacke. Cyber-crime, securities markets and systemic risk. https://www.iosco.org/research/pdf/swp/Cyber-Crime-Securities-Markets-and-Systemic-Risk.pdf.

[47] A. Waksman. A permutation network. *Journal of the ACM (JACM)*, 15(1):159–163, 1968.

[48] S. Wang. *Truthful Double Auction Mechanisms for Heterogeneous Spectrums and Spectrum Group-Buying*. PhD thesis, 2016.

[49] J. Xu, N. Vavryk, K. Paruch, and S. Cousaert. Sok: Decentralized exchanges (dex) with automated market maker (amm) protocols. *arXiv preprint arXiv:2103.12732*, 2021.

Appendix A
Bucketization Based Protocol

In Section A-A, we describe the order submission, matching and settlement for our bucketization protocol. In Section B-B we describe the privacy quantification for this scheme as evaluated in Section VII.

A. Protocol

Order submission: If the bucket start and end values were known apriori to the traders, a buyer would always choose the lower end of their intended bucket and a seller would choose the higher end, with no effect on the matching. To keep the order rates dynamic, we divide the order submission into two phases. In the first phase, traders submit orders with rate commitments. Buyers additionally submit a ZKP of range that their order rate is less than or equal to their account balance, verified in the smart contract. After T seconds or receiving M orders, whichever is earlier, the marketplace chooses random bucket start and end values for this round and publishes this on the blockchain. Deterministic random numbers can be generated within smart contracts using the block hash or timestamp as a seed [34] or using Verifiable Random Functions [23], [27]. The smart contract uses the hash of the current block as a seed to generate a deterministic random number while ensuring that traders cannot predict them. Then in the second phase, within a timeout of T' seconds, all traders must submit the bucket that their rate belongs to, along with a zero knowledge proof of range. For buy orders, the blockchain locks funds equalling the buyer's rate in an escrow using the commitment of the rate.

Matching: The marketplace performs matching using the bucket values of the orders. In case of buyers, we use the floor value of the bucket that their order price belongs to, and for sellers, we use the bucket ceiling value during matching. This ensures that any matching of counterparties that we find using our matching algorithm will definitely be feasible with respect to their private rates as well, as we are considering stricter constraints. The input to the algorithm is the set of orders sorted in increasing order of bucket values, with ties broken by timestamp, where an earlier submitted order gets precedence. Matching is performed similar to the Rialto protocol. Note that the matching algorithm matches a buyer in a higher bucket with a seller in a lower bucket and will not match orders within a bucket. This is because a buy order might have lower order rate than a sell order within the same bucket, which is an invalid match. We claim that this matching algorithm which considers buyers and sellers in increasing bucket values produces a maximum matching, among all matching algorithms with buckets and prove the same in Appendix D.

Settlement: We propose two schemes for computation of settlement rate and execution of settlement between each matched pair of buyer and seller using the traders' private proposed rates. Each matched buyer and seller is required to communicate their order rates to their counterparty, so that they are able to compute their individual final account balance after settlement.

Marketplace earns the rate difference: In this scheme, each trader settles at their respective proposed rates and the difference between the buyer and seller rates for each matched pair is paid as fees to the marketplace. The protocol proceeds as follows. In the first step, each buyer and seller encrypt their proposed rate and blinding factor used in the commitment of their order along with a digital signature, with the public key of their matched counterparty and record it on the blockchain. This helps the marketplace verify fairness of the exchange, so no party can claim that it has not received a message from the other party. Each trader decrypts their message, verifies the signature and verifies that the values match the commitment submitted by their counterparty on the blockchain. In case the values do not match, they notify the marketplace of the deviation using the digital signature as proof and the cheating party is penalized. In the second step, each trader calculates the marketplace fees as the difference of the rates and submits it to the blockchain along with a ZKP that the difference of the commitments of the buyer and seller rates on the blockchain opens to the submitted value. The marketplace verifies the proof with a smart contract on the blockchain and settles the traders by adding the seller's commitment to the seller's account balance and adding the difference value to its own account (potentially shared by the entities operating the peers of the decentralized marketplace) and removing the funds held in the escrow on behalf of the buyer.

Settlement at mean of traders' rates: In this scheme, the traders settle at the mean of their rates (the marketplace can charge a fixed fee or a percentage of the settlement rate as fee). The first step of the traders' securely exchanging their proposed rates and the blinding factors with their counterparty is the same as above.

The smart contract calculates settlement rate as the mean of commitments of buyer and seller (since PC are homomorphic), removes the funds locked in the escrow on behalf of the buyer, returns the excess funds (difference between proposed rate and settlement rate) to the buyer's account and adds the settlement funds to the seller's account.

APPENDIX B
QUANTIFYING ORDER RATE PRIVACY

A. *Rialto*

In this section, we present an analysis of the privacy achieved by Rialto, by quantifying the leakage of order rates. We also quantify the dependence of privacy on certain key parameters, such as the values for K and N, the number of top-K settled orders revealed and the total number of orders in the round, respectively.

For our analysis, we assume that the true distribution of order rates of all buyers and sellers follows a single Gaussian distribution $\mathsf{P_T}$ with mean μ_T and variance σ_T^2, unknown to all the participants. While real world order rates might have different distributions for buyer and seller rates, we make this simplifying assumption to make privacy measures uniform across all traders. We also assume that traders submit only one order in a round and knowledge is not carried across rounds of the protocol.

The marketplace reveals the top-K order rates for price discovery along with a total sorting of the orders as output by the SortingMPC. Adverseries estimate Gaussian parameters, μ_E and σ_E, from this information.

Blom [12] proposes the following equation for the expectation of the r^{th} order statistic of a Gaussian distribution, where $\alpha = \frac{\pi}{8}$ is a constant and Φ^{-1} is the quantile function of the standard Gaussian distribution.

$$\text{Blom}(r, n) = \mu_E - \Phi^{-1}\left(\frac{r - \alpha}{n - 2\alpha + 2}\right).\sigma_E$$

At the completion of any trading round, brokers can use Blom's equation using the top-K matched order rates, which are the top K order statistics, to numerically solve for parameter estimates μ_E and σ_E. Traders can additionally use their own order rate, whose r is revealed from the ranking of orders, to get more accurate estimates μ_E and σ_E. In our evaluation in Section VII, for orders generated from a true Gaussian distribution with parameters μ_T and σ_T, we use the above analysis to obtain estimates μ_E and σ_E. Finally, we measure the privacy gain (Equation 1), as the KL divergence between the Gaussian with estimated parameters (P_E) and the true Gaussian (P_T) as a percentage of the entropy of the true distribution ($H(P_T)$).

B. *Bucketization Protocol*

In the BUCKETIZATION protocol, the marketplace leaks a histogram of orders across buckets. Adversaries extrapolate the true distribution parameters from the histogram statistics and the top-K settlement rates revealed for price discovery. Privacy gain will be a function of the bucket width W and N, the expected number of orders in a round, allowing the marketplace to choose parameters depending on its privacy requirements.

Adversaries estimate each of the N orders rates in the round by sampling randomly, assuming (say) a uniform distribution within each bucket and using exact values for the top-K order rates and their own rate. They can then estimate the population (true Gaussian distribution)'s mean as the sample's mean and the population's variance by scaling the sample's variance by N to account for the smaller variance of a smaller sample.

We measure the privacy gain of the marketplace by computing the Kullback-Leibler divergence (KL divergence) of the estimated Gaussian from the true Gaussian distribution, expressed as a percentage of the entropy of the true distribution ($\mathtt{H}(\mathtt{P_T})$) as per Equation 1.

We find that privacy decreases with the number of orders, since a larger sample allows estimation of population parameters with greater accuracy. Privacy also increases with bucket width, increasing from 0.2% for width 2 to 1.2% for width 16.

APPENDIX C
AUTOMATED MARKET MAKER

Automated Market Maker (AMM) such as [8] is an automated system or algorithm to settle traders that uses a liquidity pool instead of a limit-order book for matching. Recent work [16], [49] show a rising adoption of AMM based Decentralized Exchange (DEX) for DeFi (Decentralized Finance) blockchain applications. An AMM based DEX exchanging two tokens has a liquidity pool for the token pair allowing direct exchange between the two tokens. An AMM uses a conservation function to determine the exchange rate of the asset depending on the quantity of the token assets, such as a constant product AMM where the product of the token assets is constant.

While the design and analysis of a privacy-preserving AMM DEX could be an interesting future work, we briefly describe the design of an AMM dealing with assets X and Y with token amounts T_X and T_Y respectively, which preserves the privacy of the order rate and the exchange rate.

1) **Initialization:** The liquidity pool is initialized by Liquidity Providers which provide tokens and receive pool shares in proportion to their contribution. Liquidity providers lock their tokens in an escrow and secret share their contribution among the brokers, which verify their shares. The total pool liquidity itself is hidden, maintained as secret shared between the brokers.
2) **Trade**: A trader submits an order to buy/sell tokens of Y. We can also support a trader to set a limit for the maximum/minimum tokens of X they are willing to exchange (in private, similar to Rialto). The marketplace calculates the exchange rate using MPC and checks if it matches the limits specified by the trader.
3) **Settlement:** The marketplace creates a settlement transaction to the trader, transferring the exchanged tokens in a privacy-preserving manner using PC.
4) **Price Discovery:** Finally, the marketplace could periodically reveal the pool liquidity to help with price discovery.

APPENDIX D
PROOF OF MAXIMAL MATCHING

We prove maximality of our matching algorithm using the graph theory definition of matching. We model the buyers and sellers as nodes of a bipartite graph G. There exists an edge between a buyer and seller, if the buyer's rate is no less than the seller's rate. The result of any graph matching algorithm on this bipartite graph represents a feasible matching.

An edge occurs between traders T_1 and T_2 if $T_1 > T_2$ (T_1 occurs after T_2 in the sorted list of orders).

A matching M is maximum if there does not exist any augmenting path. An augmenting path is a path between two distinct unmatched vertices where the edges are alternately in M and not in M. We claim that our matching (M_1) is maximum.

Suppose by contradiction, it is not a maximum matching. Then, there must exist an augmenting path consisting of at least 3 edges. Let this augmenting path be B_1 - S_1 - B_2 - S_2, where S_1 and B_2 are matched with each other in M_1, while B_1 and S_2 are both unmatched. Inverting the augmenting path, where B_1 is matched with S_1 and B_2 is matched with S_2, will produce a matching M_2 of greater cardinality.

We list the following facts about our matching M_1:

1) It must be that $B_1 < S_2$. Otherwise, our matching would have matched B_1 with S_2 and they would not both be unmatched.
2) Since B_2 and not B_1 is matched with S_1, it must be that $B_2 \leq B_1$. Otherwise, B_1 would have been considered earlier for matching by the algorithm and would be matched.

This implies that $B_2 < S_2$. This is a contradiction as then, an edge cannot exist between B_2 and S_2 by the construction of the graph, for it to be part of the augmenting path and matching M_2. Hence, no such larger matching M_2 can exist. Thus, our matching is a maximum matching.

APPENDIX E
ALGORITHMS

We list the algorithms used in the paper. Algorithm 1 is the swapping algorithm to swap unmatched buy orders with matched buy orders for fairness as discussed in Section V-E. Algorithm 2 is the matching algorithm used by the marketplace to match buyers and sellers. Algorithms 5, 6 and 7 show the sequence of steps at the marketplace, a trader and a broker respectively for the Rialto protocol. Algorithms **??** and **??** show the sequence of steps for the marketplace and a trader respectively for the BUCKETIZATION PROTOCOL.

Algorithm 1 Swapping algorithm for fairness

```
1: procedure SWAPPING(MatchedOrders, SortedOrders)
2:     while Exists unmatched buyer with higher price than a matched buyer do
3:         swap highest price unmatched buyer replacing highest price matched buyer lesser than it
4:     end while
5: end procedure
```

Algorithm 2 Matching algorithm

```
1: procedure MATCHING(SortedOrders)
2:     BuyIdx, SellIdx := 0, 0
3:     BuyOrders, SellOrders := SortedOrders
4:     MatchedOrders := []
5:     while BuyIdx $<$ len(BuyOrders) and SellIdx $<$ len(SellOrders) do
6:         BuyOrder := BuyOrders[BuyIdx]
7:         SellOrder := SellOrders[SellIdx]
8:         if (BuyOrder $\geq$ SellOrder) then
9:             MatchedOrders.APPEND((BuyOrder, SellOrder))
10:            BuyIdx := BuyIdx + 1; SellIdx := SellIdx + 1
11:        else
12:            BuyIdx := BuyIdx + 1                ▷ Cannot match
13:        end if
14:    end while
15:    return MatchedOrders
16: end procedure
```

Algorithm 3 Bucketization - Marketplace Smart Contract

```
1: do in background
2:     Order$_i$ := (Account$_i$, RateComm) <- RECV ORDER FROM TRADER $i$
3:     UnmatchedOrders.APPEND(Order$_i$)
4: end

5: function TRADINGROUND(UnmatchedOrders, BucketWidth)
6:     Wait for T seconds; then Orderbook := [UnmatchedOrders]
7:     Buckets := CHOOSE_BUCKETS(BucketWidth)
                                        ▷ Choose random values
8:     SEND BUCKETS TO TRADERS(Buckets)
9:     repeat
10:        (Id, Bucket$_i$, Proof$_i$) <- RECV FROM TRADER $i$
11:        VERIFY_RANGE_PROOF(Orderbook[Id], Bucket$_i$, Proof$_i$)
12:        Orderbook.UPDATE(Id,Bucket$_i$)
13:    until T' seconds or all proofs received
14:    SortedOrders := SORT(Orderbook, ascending)
15:    MatchedOrders := MATCHING(SortedOrders)
                                        ▷ Algorithm 2
16:    for (BuyOrder, SellOrder) in MatchedOrders do
17:        (Fees, Proof') <- RECV FEES FROM BUYER,SELLER
18:        FeesComm := BuyOrder.Comm $-$ SellOrder.Comm
19:        VERIFY_PROOF(Fees, FeesComm, Proof')
20:        DEBIT(BuyOrder.Account, BuyOrder.RateComm)
21:        CREDIT(SellOrder.Account, SellOrder.RateComm)
22:        CREDIT(Marketplace.Account, Fees)
                                        ▷ Credit to marketplace
23:    end for
24:    UnmatchedOrders := Orderbook $-$ MatchedOrders
25: end function
```

Algorithm 4 Bucketization - Trader

```
1: function TRADE(Rate, Blinding, Account)
2:     RateComm := PEDERSEN_COMMITMENT(Rate, Blinding)
3:     Id := SEND ORDER TO MARKETPLACE(Account, RateComm)
4:     Buckets <- RECV BUCKETS FROM MARKETPLACE
5:     Bucket := RATE_TO_BUCKET(Rate, Buckets)
6:     Proof := GENERATE_RANGE_PROOF(Rate, RateComm, Bucket)
7:     SEND TO MARKETPLACE(Id, Bucket, Proof)
8:     MatchedTrader <- RECV MATCHING FROM MARKETPLACE
9:     $M$ := (Rate, Blinding)
10:    $\sigma$ := DIGITAL_SIGNATURE($M$)
11:    $E$ := ENCRYPT($M$, $\sigma$)
12:    $E'$ := EXCHANGE RATES WITH MATCHEDTRADER($E$)
13:    (Rate', Blinding', $\sigma$') := DECRYPT($E'$)
14:    VERIFY_SIGNATURE($\sigma'$)
15:    Fees := Rate - Rate'
16:    Proof' := GENERATE_PROOF(Fees, Blinding - Blinding')
17:    SEND FEES TO MARKETPLACE(Fees, Proof')
18: end function
```

Algorithm 5 Rialto - Marketplace Smart Contract

```
1: do in background
2:     Order$_i$ := (Account$_i$, ShareComms, BalProof, OrderType) <- ORDER FROM TRADER $i$
3:     Order$_i$.RateComm := RECONSTRUCT(ShareComms)
4:     if Order$_i$.OrderType == "BUY" then
5:         VERIFY_RANGE_PROOF(Order$_i$.BalProof)
6:     end if
7:     UnmatchedOrders.APPEND(Order$_i$)
8: end

9: function TRADINGROUND(UnmatchedOrders)
10:    Wait for T seconds; then Orderbook := [UnmatchedOrders]
11:    SortedOrders := BROKERS_SORT(Orderbook)
                                        ▷ Event trigger to brokers
12:    MatchedOrders := MATCHING(SortedOrders)
13:    Fees := BROKERS_SETTLE(MatchedOrders)
                                        ▷ Event trigger to brokers
14:    for (BuyOrder, SellOrder) in MatchedOrders do
15:        DEBIT(BuyOrder.Account, BuyOrder.RateComm)
16:        CREDIT(SellOrder.Account, SellOrder.RateComm)
17:    end for
18:    CREDIT(Marketplace.Account, Fees)
19:    UnmatchedOrders := Orderbook $-$ MatchedOrders
20:    BROKERS_SHUFFLE( )       ▷ Event trigger to brokers
21: end function
```

Algorithm 6 Rialto - Broker

```
1: do in background
2:     OrderShare[Id] := (Id, RateShare, BlindingShare, Account) <- RECV SHARE FROM TRADER
3:     VERIFY_RATE_SHARE(RateShare, Orders[Id])    ▷ Read Order from Blockchain
4:     VALIDATE(RateShare)    ▷ Rialto+ (Sec V-D)
5:     VALIDATE(BlindingShare)    ▷ Rialto+
6: end

7: function BROKERS_SORT(Orderbook)
8:     RateShares := [OrderShare[Id].RateShare for Id in Orderbook]
9:     return INVOKEMPC(SortingMPC,RateShares)
10: end function

11: function BROKERS_SETTLE(MatchedOrders)
12:     for (BuyOrder, SellOrder) in MatchedOrders do
13:         BuyerShares.APPEND(OrderShare[BuyOrder.Id].RateShare)
14:         SellerShares.APPEND(OrderShare[SellOrder.Id].RateShare)
15:     end for
16:     return INVOKEMPC(SettlementMPC,(BuyerShares, SellerShares))
17: end function

18: function BROKERS_SHUFFLE
19:     BlindingShares := [OrderShare[Id].BlindingShare for Id in Orderbook]
20:     Accounts := [OrderShare[Id].Account for Id in Orderbook]    ▷ Read new account balance commitments from ledger
21:     return NewAccounts := INVOKEMPC(ShuffleMPC, (BlindingShares, Accounts))
22: end function
```

Algorithm 7 Rialto - Trader

```
1: function TRADE(Rate, Account, OrderType)
2:     RateShares := SPLIT(Rate, M)
3:     Blinding := RANDOM( )    ▷ Acc. Bal. Re-randomzation
4:     BlindingShares := SPLIT(Blinding, M)
5:     RateComms := [PEDERSEN_COMMITMENT(Share) for Share in RateShares]
6:     BlindingComms := [PEDERSEN_COMMITMENT(Share) for Share in BlindingShares]
7:     if OrderType == "BUY" then
8:         BalProof := GENERATE_RANGE_PROOF(Rate, Account)
9:     end if
10:     Id := SEND ORDER TO MARKETPLACE(Account, ShareComms, BalProof, OrderType)
11:     for Broker b in [1,M] do
12:         SEND RATE SHARE TO BROKER(Id, OPEN(RateComms[b]), OPEN(BlindingComms[b]), Account)
13:     end for
14: end function
```