XChange: A Generic Blockchain Mechanism for Trading at Scale

Martijn de Vos and Johan Pouwelse
Delft University of Technology

Abstract
Decentralized marketplaces enable traders to exchange value without a central authority. Blockchain technology is increasingly being used to deploy decentralized marketplaces. However, the architecture of traditional blockchain-based platforms is not suitable to facilitate generic trading. The cardinal issue is the requirement to establish a global consensus on all market transactions stored on the blockchain ledger. This severely limits scalability and results in intolerable delays before new orders are confirmed by the network.

We present XChange, a generic blockchain mechanism for trading at scale. Our mechanism is specifically designed for segmented markets, i.e., based on geography or demography. We use a scalable blockchain fabric that stores all orders and trade on a distributed ledger. By relying on fraud detection instead of fraud prevention, we avoid the requirement of having global consensus. Our experimental evaluation shows that: (1) XChange offers a significant improvement in terms of scalability, compared to state-of-the-art decentralized markets; (2) even the most challenging kind of fraud (transaction hiding) can be detected within mere seconds; and (3) XChange is resistant against crashes of deployed instances.

CCS Concepts → Computer systems organization → Peer-to-peer architectures;

Keywords Blockchain, Decentralized Markets, Trading, Scalability, Global Consensus

1 Introduction
For decades, the ability to facilitate trade at a global scale has been at the heart of many established companies operating on the Internet. In 1995, Craigslist already offered an unmoderated mailing list where strangers could negotiate, meet, and trade. A few years later, eBay formalised the concept of online trading by introducing a reputation system where buyers and sellers rate each other. More recently, leading companies acting within the sharing economy, deployed global marketplaces for ride-hailing (Uber), accommodation (Airbnb), freelance labour (TaskRabbit) and many other services. We argue that the decentralization is next step, the ability to establish trade and exchange value without trusted market operator.

This work presents XChange, a generic, blockchain-based architecture to build decentralized marketplaces. Our architecture does not need any central market operator to bring buyers and sellers together. Our middleware is capable of trading any digital asset or service. Unlike related blockchain-based markets, XChange does not require network-wide validation of market activity (like creating orders). As we will show, this yields a significant improvement in terms of scalability, compared to the state-of-the-art.

Most digital marketplaces are operated by a central market authority, responsible for defining and controlling the environment in which trade is established and executed [44]. Having a single market operator in control has desirable properties. Trade usually takes place in a well-defined environment and proceeds in an organized manner. Yet, centralization of market authority leads to a few deficiencies in general. First, reputation accumulated by traders is often locked to a single platform and cannot easily be transferred elsewhere. This results in fragmentation of one’s trading history. Second, centralized architectures tend to be less resistant against infrastructure failures and targeted attacks (for instance, a Denial-of-Service attack [28]). Finally, deployed business logic of (commercial) market operators is rarely open for inspection by traders. This motivates market authorities to exploit their eminent position. For example, it is suspected that Uber manipulates ride prices with their dynamic pricing mechanism (surge pricing) [8].

Cryptocurrency marketplaces highlight various issues caused by centralized architectures [29]. These markets are founded to facilitate the need to trade different cryptocurrencies and digital tokens. While a few of these markets process transactions worth millions of dollars on a daily basis, many market operators lack the knowledge or resources to quickly scale up their infrastructures to meet increasing demand. More than once, the fragile infrastructure of centralized cryptocurrency marketplaces leads to poor trading experiences, prolonged platform unavailability, and even the inability to withdraw funds from digital wallets [7]. Over the past years, several high-profile breaches and hacks revealed that wallets containing cryptocurrencies are not always securely stored by market operators. The largest hack since the inception of cryptocurrencies took place in 2014, when hackers compromised the security of Mt. Gox and stole Bitcoin, worth around $460 million at that time [25].

The shortcomings of centralized marketplaces motivate the design of decentralized marketplaces. On these platforms, traders control funds themselves, instead of having these managed by a central party. Thus, decentralization increases robustness against large-scale hacks, since wallets are not stored in a single location. However, other issues of centralized infrastructures are not automatically dismissed by decentralization. In fact, new challenges have to be addressed, including fraud management and dispute resolution, secure accounting of activity, and effective dissemination of market information. Various decentralized marketplaces address these issues by building on blockchain technology, where market activity is fully or partially stored on a distributed ledger [40].

In this work, we first analyse issues when relying on traditional blockchain technology to deploy decentralized marketplaces. We then improve upon an existing blockchain ledger that differs significantly from traditional blockchain architectures: TrustChain. It is demonstrated how TrustChain is used to host a fully decentralized marketplace, capable of storing all market activity. TrustChain aims to detect fraud, instead of preventing it. We prove that this is a viable design decision and that even the most extensive kind of fraud (transaction hiding) can be detected and managed within seconds. We implement a fully functional marketplace and evaluate it using real-world stock trading traces of the Apple stock.

Our contributions are four-fold:
1. A novel, scalable architecture and blockchain design to devise decentralized marketplaces, based on locality and segmentation instead of global consensus (Section 3 and 4).
2. Anti-fraud measures with experimentally proven detection times within seconds (Section 4.5).
3. A fully functional, open source implementation of XChange, our devised trading mechanism (Section 5).
4. Experimentation around scalability and fault tolerance, comparing XChange with the state-of-the-art (Section 6 and 7).

2 Problem Statement

The blockchain is a distributed ledger on which transactions are stored. The state of the chain is maintained by individuals, without involvement of trusted third parties. A blockchain consists of blocks, and each block can contain transactions. The size of an individual block is often limited. New blocks can only be appended to the end of the blockchain and it is computationally impractical to alter existing blocks in the ledger. This tamper-proof property is realised by inclusion of a hash that is derived from the prior block in the chain. Most blockchain ledgers are public and everyone can verify correctness and consistency of their blocks.

To secure the blockchain, a global consensus mechanism is often used. This mechanism coordinates which user is able to append new blocks to the chain and resolves conflicts when there are two different (valid) versions of the blockchain ledger (also called forks). It builds agreement about the current state of the blockchain and ensures that each transaction in the chain is valid and agreed on by the network. Establishing global consensus is a key requirement to manage digital currencies without central authority and as such, it is an essential component of many blockchain architectures. However, we identify four issues when building marketplaces on traditional blockchain ledgers, caused by the need for global consensus.

The first issue is that reaching global consensus is expensive in terms of resource usage. It impacts the overall efficiency of the ledger, namely the rate at which new blocks and transactions can be appended to the chain. The maximum throughput of blockchain applications is often constant. Gervais et al. found that traditional consensus mechanisms in blockchains, based on computing power, are limited to around 60 transactions per second at most [16]. Considering throughput needed to process payments world-wide, many blockchain applications are not scalable enough yet to provide viable alternatives for financial industry (leading credit card companies process thousands of transactions every second [45]).

Second, it can take a considerable amount of time before new orders submitted by traders are included within a block on the chain. Even though the interval between block creation can be as low as a few seconds, it is still unsuitable for situations that require near-instant confirmation of market activity. Examples of such situations include real-time bidding and high-frequency trading.

Third, blockchain applications reward users for their participation in the global consensus mechanism. This reward is covered by users who initiate transactions. While traders also have to pay transaction costs on most centralized markets, trading in large volume is costly and unattractive.

Finally, it is questionable whether traditional blockchain architectures with need for global consensus are appropriate to effectively facilitate real-world trading. In particular, trading often proceeds on segmented markets. Stock trading markets operate highly isolated from each other, based on geographical location. Another example is local energy trading, where interaction is often limited to a neighbourhood or geographical district (since it is inefficient to transmit bought energy over a large distance) [27]. We believe that establishing global consensus on aggregated information, generated by traders on segmented markets, is unnecessary. The challenge is to offer the same security with a less costly consensus model.

Based on these four issues, this work answers the following research question: how can we devise and build a generic, fully decentralized mechanism to trade assets and services, both on local and global scale, but without global consensus? Our mechanism must be able to bring traders together in a fully decentralized manner, without trusted intermediaries. On the other hand, it should be generic enough to facilitate trade beyond cryptocurrencies and digital tokens, as existing blockchain-based markets offer.

3 XChange Architecture

Based on our research question, we identify all components required for a decentralized market and devise a high-level architecture of XChange, our trading mechanism. This architecture is presented in Figure 1, and also shows which components exchange data with each other. All components displayed in the figure will be covered in this section, except for the distributed ledger and fraud detection service. These components will be explained in Section 4.

3.1 Trading API

First we discuss the Trading API, which acts as the primary interface between users and the XChange mechanism. It allows traders to interact with or manipulate the state of the market and provides them with the following functionalities:

- Fetch open orders on the market.
- Submit new orders to the market.
- Cancel previously submitted orders.
- Interact with available wallets (see Section 3.5).

We devised the Trading API as a RESTful API. Developers can build applications for end users with it. Modification of the market state is done by sending HTTP POST requests to the API and fetching available information by sending HTTP GET requests. While the Trading API is not an essential component of XChange itself, it provides a convenient interface to interact with the market.

3.2 Local Order Book

The local order book in XChange stores discovered orders created by traders. First, we define how an order is represented in XChange. Next, we discuss the functionality of a local order book.

Buy and sell orders. An order indicates the willingness of a trader to buy or sell specific assets or to exchange services with others. Orders that aim to sell particular assets or services are referred to as ask orders whereas orders that aim to buy them are called bid orders. The set of open ask orders represents the current supply and the set of open bid orders represent the current demand. Usually, orders contain timestamps of creation and expiration (indicating when the order expires and when it will be removed from the market if not completed yet). Open orders can often be cancelled by their creators at any time, and are then removed from the market.

On markets where financial instruments are traded (like stocks or currencies), an order is placed for a specific price and amount. In other market types, an order might be expressed alternatively, with other rules and attributes. For example, within a ride-hailing marketplace like Uber, the willingness to act as a driver can be mapped to a sell order and a transportation requests by a passenger can be interpreted as a buy order. Such orders also incorporate the physical location of the passenger or driver (i.e. in latitude and
When XChange discovers that the status of an order has changed, trade (the exchange of assets or services) is established by the order matching engine. The order matching engine continuously tries to match buy and sell orders in a local order book. Whether buy and sell orders match, depends on the attributes of these orders. In a decentralized environment, without central market operator, efficient and effective matchmaking is harder, due to the heterogeneous hardware of individuals. Moreover, decentralized matchmaking is usually performed by a group of users and they might have to establish matches based on partial or incomplete market information. This potentially decreases matchmaking efficiency [6].

We now elaborate on the process of matchmaking in XChange. By default, every trader within the XChange network performs matchmaking and attempts to match new incoming orders with existing ones in their local order book. We call this third-party matching. Traders that perform matching, are called matchmakers. When a matchmaker finds matches in the local order book, it fully or partially reserves the matched orders. This ensures that the reserved fraction of an order will not be considered for consecutive matching operations, until it is released again. To support this, we extend order book entries with an attribute that represents how

**Figure 1.** A high-level architecture overview of our trade mechanism XChange. Arrows between two components indicate they exchange data. We represent related components by the same colour.

longitude coordinates). Similarly, sell orders on Airbnb are created by property owners who want to rent out their accommodation and buy orders are created by tourists, looking for a place to stay.

**Order Books.** In XChange, each trader organizes all known orders inside a local order book. An order book lists the specific assets or services that are being bought or sold within a market. It provides traders with a convenient view on the current supply and demand. Depending on the market and assets being traded, the order book sometimes reveals the identity of a trader behind an open order, or their reputation scores (for instance, Airbnb shows the reputation of hosts). Figure 2 shows an example of an order book used in a stock trading market, with three bid orders and three ask orders. When presenting an order book to traders, ask and bid orders are usually sorted, i.e. based on their price. For orders with a price, ask orders are sorted ascending on price and bid orders descending on price in the order book (the best orders are presented first). The set of all ask or bid orders at a specific price is called a price level. Referring to Figure 2, the ask price level $149.58 consists of two orders, created by traders D and B respectively.

Every trader in XChange maintains a local order book with all open orders. As we will elaborate in Section 4, XChange explores market data stored on the distributed ledger (the blockchain) and aims to find orders created by other traders. When XChange discovers a new order, the validity of the order is checked. The validation function will verify if the order has not expired yet and whether it is valid with respect to the rules of the trading environment. Only if an order is deemed valid, it will be inserted in a local order book. Expired orders are automatically removed from the local order book. When XChange discovers that the status of an order has changed, the corresponding order book entry is updated accordingly.

We do not particularly focus on pricing mechanisms and decentralized price discovery in this work. Instead, we aim for our trading architecture to be independent of the type of assets or services being traded. Various components, like the order matching engine and the local order book, could optionally be adapted to better support operations on specific types of markets.

### 3.3 Order Matching Engine

Trade (the exchange of assets or services) is established by the order matching engine. The order matching engine continuously tries to match buy and sell orders in a local order book. Whether buy and sell orders match, depends on the attributes of these orders. In a
much of the order has been reserved. Order reservations will only
be released when a trade has successfully been completed or has
failed. When a matchmaker establishes one or more matches for
a new incoming order, the creator of this incoming order is notified.
The matchmaker sends a message to this order creator, containing
specifications of the found match. We name this message a match
notification. Traders that receive an incoming match notification,
can either accept or decline them. Trade now proceeds between two
traders, without involvement of the matchmaker (see Section 3.4).
On completion or failure of the trade, the matchmaker is informed
and updates their order book entries accordingly (by releasing the
full reserved quantity or a fraction).

Note that matching in this fashion can lead to the situation where
an order creator receives two or more match notifications within a
short period of time, from different matchmakers. A trader should
now determine which match notifications it accepts. For instance,
a basic policy is to only accept the first few incoming match noti-
fications. While this likely results in faster order fulfillment times,
a trader could receive a match notifications with more beneficial
matches from other matchmakers right after accepting some match
notifications. Also, with this policy matchmakers can purposefully
send sub-optimal matches to specific traders and decrease overall
efficiency of the market. Alternatively, traders can accumulate in-
coming match notifications during a pre-defined period and accept
the optimal matches, and decline all other match notifications. We
assume that the exact policy to deal with multiple incoming match
notifications, is defined by application-specific logic.

3.4 Trade Execution Engine

The trade execution engine manages and controls the trading pro-
cedure between two or more trading parties. In XChange, a trade
proceeds in three phases: negotiation, clearing and settlement.

Negotiation. After accepting an incoming match notification,
negotiation starts between two traders. A trader directly contacts
other traders whom it is matched with by sending a trade proposal.
A trade proposal contains the desired asset types and quantities
this trader wants to exchange with the counterparty. On receiving
a trade proposal, a trader verifies whether his order is still open (the
order could already have been fulfilled by a match from another
matchmaker or have been expired). The trade proposal is now either
accepted or declined, or a counter proposal is sent. When a trader
accepts a trade proposal or a counter proposal, the negotiation
phase is completed and the clearing process starts.

Clearing. Clearing proceeds before assets or services are ex-
changed. Objectives of the clearing phase includes checking whether
both parties have enough balance to complete the upcoming trade
and whether the trade complies with local and international reg-
ulation. In centralized markets, clearing is often performed by a
trusted third party, also called a central clearing house.

In XChange, clearing proceeds as follows. First, traders exchange
the necessary information to ensure assets can successfully be trans-
ferred to each other. This includes the address on which traders
expect assets to be received, for instance, a bank account number
(when trading for fiat money) or a physical address (when trading
goods). Both parties now construct and sign a digital agreement,
containing specifications of the upcoming trade. This agreement
commits the trading parties to the trade and is stored on the dis-
tributed ledger used by XChange (see Section 4.6). This completes
the clearing phase of a trade, and settlement is initiated next.

Settlement. During the settlement phase, traders actually ex-
change assets or services with each other. It is the process whereby
parties discharge their contractual obligations to pay each other.
Settlement might rely on other trusted parties, for instance, pay-
ment providers or shipping companies. Depending on the assets
being traded, duration of settlement might range from millisec-
onds (i.e. when transferring money within the same bank) to a few
days or longer (i.e. when sending physical goods to foreign coun-
tries). At a minimum, each trade between traders $A$ and $B$ involves
two asset transfers; one transfer from $A$ to $B$ and one from $B$
to $A$. We capture each asset transfer operation in a transaction,
called a proof-of-delivery and store it on our distributed ledger. The
proof-of-delivery is digitally signed by both trading parties. The
two signatures in a proof-of-delivery ensure that assets have been
successfully sent by one party and received by the other during
asset transfeff. A trade is completed by constructing, signing and
storing a digital agreement that indicates all expected assets have
been sent and received during a particular trade.

3.5 Wallets and Settlement Providers

The actual transfer of assets to others is performed by two compo-
nents in our architecture: wallets and settlement providers. Different
wallets and settlement providers can be defined by developers.

Wallets. A wallet is a primitive to manage assets, and is the
gateway to interact with settlement providers. Wallets can store any
(digital) asset, like cryptocurrencies, fiat money in a bank account
or digital tokens. Wallets expose functionality to transfer assets, to
query available asset balance and to fetch historical transactions.
We assume individual wallets are uniquely identified by an address.
Examples of wallet addresses are public keys (for wallets that store
cryptocurrency), bank account numbers (for wallets that store fiat
money), or email addresses (for wallets that interact with PayPal).

Settlement Providers. The responsibility of settlement providers
is to arrange the transfer of assets to others during a trade. Examples
of settlement providers include cryptocurrency networks, finan-
cial institutions or shipping companies. Wallets interact with these
settlement providers and instruct them to transfer assets to other
users. We should note that it is also possible to transfer tokens to
others by using our distributed ledger (see Section 4.3). Transferring

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Order ID</td>
<td>Trader</td>
<td>Amount</td>
<td>Price</td>
<td>Price</td>
<td>Amount</td>
<td>Trader</td>
<td>Order ID</td>
<td></td>
</tr>
<tr>
<td>#7825</td>
<td>A</td>
<td>800</td>
<td>$149.54</td>
<td>$149.57</td>
<td>294</td>
<td>B</td>
<td>#7829</td>
<td></td>
</tr>
<tr>
<td>#7826</td>
<td>C</td>
<td>439</td>
<td>$149.53</td>
<td>$149.58</td>
<td>400</td>
<td>D</td>
<td>#7828</td>
<td></td>
</tr>
<tr>
<td>#7837</td>
<td>D</td>
<td>830</td>
<td>$149.51</td>
<td>$149.58</td>
<td>300</td>
<td>B</td>
<td>#7820</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2. Example of an order book used in a stock trading market. The order book contains six open orders, three bid (buy) orders and
three ask (sell) orders. Bid orders are sorted descending on price and ask orders are sorted ascending on price.
such tokens does not involve external settlement providers and the transfer operation can be executed within XChange itself.

4 Scalable Blockchain Accounting

Throughout the discussion in Section 3, we have assumed that there is a distributed ledger, capable of storing market information. Building upon a blockchain ledger as used in Bitcoin and Ethereum might seem like an informed decision. However, in Section 2 we argue such ledgers are unsuitable for trading, due to the limitations imposed by the global consensus mechanism. Instead, we improve upon a specialized and scalable blockchain fabric that records transactions between individuals. In Section 4.1 to Section 4.5, we present how this blockchain fabric is organized. Finally, in Section 4.6 we show how this ledger is used to store orders and trade.

4.1 Traditional Blockchain Fabrics

Widely used blockchain applications like Bitcoin and Ethereum deploy a single ledger, shared and maintained by the network itself. Figure 3 shows a simplified representation of such a blockchain, with three blocks. Each block contains one or more transactions, a nonce (used by the consensus mechanism) and the root hash of the merkle tree with all hashes of transactions in the block.

Figure 3. The traditional blockchain data structure, as used by Bitcoin and Ethereum. Each block contains a number of transactions, a nonce (used by the consensus mechanism) and the root hash of the merkle tree with all hashes of transactions in the block.

In most blockchain applications, the consensus mechanism selects a leader periodically, who is allowed to extend the blockchain with exactly one block. In the Bitcoin network, a new block is appended roughly every ten minutes [30]. Users participating in the consensus mechanism are called miners. For Bitcoin and Ethereum, each miner in the network attempts to solve a computational puzzle and the first one to find a solution, can append a new block (this is called Proof-of-Work consensus). The difficulty of the computational puzzle grows with the overall computing power of the network. Other consensus mechanisms select a leader based on the user’s stake in the network (this is called Proof-of-Stake consensus) [3]. Without such a consensus mechanism, the network would be void of agreement on the current state of the blockchain.

4.2 Sharding

Scalability is a desired attribute of blockchain fabrics [42]. One proposed mechanism to improve scalability is sharding, where the entire state of the blockchain network is partitioned into shards [5]. Each shard maintains its own state and transaction history. Nodes in the network process transactions only for certain shards. Organization of these shards could be based on address space, geographical location or different currency types represented on a blockchain.

Sharding seems like a viable mechanism to facilitate trade on segmented markets (as discussed in Section 2). By dividing market operations in separate shards, scalability is increased. However, there are various issues with this. First, sharding is a complex mechanism with an extensive design space and numerous open questions [10]. Blockchain developers should make sure that shards cannot easily be taken over by malicious users. Second, communication between shards is a non-trivial problem. Third, there is no mature and proven blockchain platform that exclusively increases scalability through sharding, to the best knowledge of the authors.

4.3 TrustChain: A Scalable Blockchain Ledger

Otte et al. proposed to use a mechanism, similar to sharding, to devise a scalable blockchain: TrustChain [34]. In TrustChain, consensus is only reached between interacting parties instead of globally. TrustChain is specifically designed to store bilateral interactions. Figure 4 illustrates how a transaction is recorded between two interacting users in TrustChain. Figure 4a shows the record of a single transaction (Tx), initiated by user A. A transaction is a generic description of any interaction between users, for instance, making an agreement or recording a trade. Both transacting parties digitally sign the transaction they are involved in by using any secure digital signing algorithm. These signatures are included in the transaction record and ensure that participation by both parties is irrefutable. It also confirms that both parties agree with the transaction itself. Digital signatures can be effectively verified by others. After all required signatures have been added to a record, the transaction is committed to the local databases of the two interacting parties.

Security of stored records is improved by linking them together, incrementally ordered by creation time. In particular, each transaction record is extended with a description (hash) of the prior record and becomes a block in the resulting chain. Each block has a sequence number that indicates its position in the chain, and has exactly one transaction. This yields the blockchain structure as shown in Figure 4b. As a result, each user maintains their own local chain which contains all transactions in which they have participated. This makes TrustChain different from the blockchain structure as shown in Figure 3 (where the network maintains a single ledger).

The blockchain in Figure 4b is void of any control, outside for the user maintaining that chain. This allows users to modify records in their local chain without guarantees of being detected by others. To elaborate this, users are able to reorder blocks in their local chains since validity can easily be restored by recomputing all block hashes. In blockchain applications that deploy a single ledger, the global consensus mechanism prevents this kind of modification. TrustChain relies on a more efficient solution: each block is extended with an additional hash that describes the prior block in the chain of the transaction counterparty. This is presented in Figure 4c. Each block now has exactly two incoming and two outgoing pointers, except for the last block in a local chain (which only has two incoming pointers). Fraud, like reordering or removing blocks, can now be detected and proven by the other party in a transaction.

When two parties transact and create a block, their chains essentially become entangled. When users initiate more transactions with others, it leads to the directed acyclic graph (DAG) structure as shown in Figure 5. Figure 5 shows seven blocks, created by seven
According to Otte et al., TrustChain is designed to scale [34]. How-
agrees with the transaction, he signs the block partition created by
content and his digital signature. User
ated to the chain of one transaction party. First, user
partition a block in two parts and each block partition is commit-
features of the transaction counterparty is required before a new block
of a block is pending. The main issue here is that the digital signa-
creation at once. No other blocks can be created while construction
limits scalability of their approach. It also enables an attack where
block also includes all signatures in the prior block). This severely
controls a new block, it will be verified by a validation method. This
system that is influenced by applications built on top of the ledger.
users are valid and consistent. Users continuously explore
network and request blocks in the chain of others. The rate
participation in them. Alternatively, users might be tempted to alter
on the distributed ledger is to hide past transactions or denying
users in the network.
architecture, is the
architectures, is the
net-work in order to prove fraud attempts.
users in the network.
chains of others, is a parameter of the
the TrustChain fabric to host a digital currency with limited
supply requires additional work since block creation is uncoordi-
nated. For our work, we use TrustChain to store market activity
and asset transfer is done by settlement providers (see Section 3.5).
Tokens stored on the TrustChain ledger can be traded on XChange.

4.4 Improving Scalability

One application of TrustChain is the storage of basic tokens. Using the TrustChain fabric to host a digital currency with limited supply requires additional work since block creation is uncoordinated. For our work, we use TrustChain to store market activity and asset transfer is done by settlement providers (see Section 3.5). Tokens stored on the TrustChain ledger can be traded on XChange.

4.5 Fraud Detection

Users can be inclined to attempt fraud by targeting the structure of the TrustChain ledger. An incentive to rearrange information stored on the distributed ledger is to hide past transactions or denying participation in them. Alternatively, users might be tempted to alter existing transactions for personal enrichment or to gain a better standing in the network.

To detect fraudulent behaviour of users, every network participants runs a fraud detection service that verifies whether the chains of other users are valid and consistent. Users continuously explore the network and request blocks in the chain of others. The rate at which users explore the chains of others, is a parameter of the system that is influenced by applications built on top of the ledger. Faster exploration leads to a more complete history of interactions that other users participated in, at the cost of increased bandwidth and CPU usage to query and verify these blocks. When a user discovers a new block, it will be verified by a validation method. This method includes a check whether the digital signatures match with the transaction content and whether all hashes are correctly computed. An invalid block can be ignored or disseminated throughout the network in order to prove fraud attempts.

One of the most challenging attacks in traditional blockchain architectures, is the double spend attack. During a double spend attack, a previously valid transaction on the blockchain is reverted,
We are now ready to show how we can securely store orders and whether an order is valid and contains the required attributes. In this work, we assume that the role of a witness is fulfilled by other traders in order to keep our design fully decentralized. However, a witness can also be operated by a third party (possibly for a fee). Depending on the order attributes, a witness could also verify whether the order creator has sufficient funds to cover the order. Similar to order creation, cancellation of open orders proceeds by creating a transaction with a witness, with the cancel transaction type. We assume that each trader randomly selects witnesses in the network. In Section 8, we explain how to deal with malicious witnesses.

Local order books in XChange (see Section 3.2) are synchronized by gossiping TrustChain blocks around the network (push) and requesting blocks in the chains of others (pull). New traders can bootstrap their local order book by exploring the chains of other traders, working from the latest block in each chain towards the genesis block. Requesting blocks of other traders at a faster rate, results in quicker reconstruction of the global order book and more complete market information. On the other hand, it increases bandwidth usage and system load to verify validity of blocks.

**Storing Trades.** Section 3.4 specifically showed how two traders exchange assets in XChange. Recall that an agreement to trade during the clearing phase is established first, prior to settlement. We store this agreement as a transaction in TrustChain, with the init type. Proof-of-deliveries are stored as transactions with delivery type, only signed by traders when assets have been sent by one party and received by the other. A trade is completed by recording a transaction with the done type on the local chains of traders.

## 5 Implementation

We now describe the implementation of XChange.

**XChange Implementation.** We implemented the XChange trading mechanism in the Python 2.7 programming language. Our implementation includes all components presented in Figure 1, with low coupling between them. It spans 3.091 lines of code of which 30.6% is overlay network logic. For persistent storage of the local order book, we use the SQLite library [39]. We have built a basic RESTful API for traders to interact with XChange (see Section 3.1). We have also implemented a Bitcoin wallet, based on the Electrum library, that developers can utilize for Bitcoin trading. Our order matching engine uses a time-price matching strategy by default, although developers can implement custom strategies. The XChange implementation is published on GitHub\(^1\).

**Blockchain Fabric.** This work builds upon the existing implementation of TrustChain, published by Otte et al [34]. We extended the TrustChain implementation to support concurrent block creation (see Figures 4d), improved the fraud detection mechanism and expanded the test suite so the tests cover our improvements. The improved implementation of TrustChain is published on GitHub\(^2\).

**Networking.** We decided to build XChange on top of an existing networking library\(^3\). Our motivation for this, is that TrustChain is built upon the same networking library. This library provides us functionality to quickly devise decentralized overlay networks and has built-in support for authenticated network communication, custom message definition and UDP hole punching.

**Quality Assurance.** We aim for our work to be used by the financial industry eventually. Deploying robust blockchain mechanisms used by millions, is non-trivial. In 2016, an attacker found

---

1. Omitted for double-blind review.
2. Omitted for double-blind review.
3. https://github.com/qstokkink/py-ipv8
6 Experimental Setup

We now present the setup of our experimental evaluation. We also only includes new orders that differ we use the sample data provided by them, which contains times-
dynamic arrival rate of orders. The LOBSTER organization publishes
data to verify the correctness of our implementation under a dy-
complement the increased load on the matching engine.

We pick the datasets that describes trading activity of the Apple (AAPL) stock symbol since this data provides us with the highest
variation in complexity. We use the datasets with price level update ranges (complexity) 1, 5, 10, 30 and 50. For each dataset, we collect
messages generated during the first five minutes (right after the opening of the trading venue). Since our dataset lacks the identity
of the trader behind a new order, we assign new incoming orders to all running XChange instances in a round-robin fashion.

6.2 BitShares and Waves

We compare XChange with related blockchain-based marketplaces. We now describe how these platforms operate and how we configure these applications during our experiments.

**BitShares.** BitShares enables users to issue, manage and trade their own assets, which are stored on a single blockchain [36]. To coordinate block creation, BitShares uses the Delegated Proof-of-Stake (DPoS) consensus mechanism [22]. DPoS utilizes approval voting to decide on a committee of so-called witnesses. Witnesses are able to append a new block to the blockchain (produce a block), in a round-robin fashion, and are rewarded by the sum of transaction fees in a block they have added to the chain. If a witness acts malicious, i.e. by deliberately failing to produce a new block, the witness will eventually be removed from the committee by stakeholders. All orders submitted by traders are stored on the BitShares blockchain. Matching of orders proceeds by a deterministic algorithm. The matches themselves are not stored on the blockchain, to improve efficiency (but they can be deduced by users).

We compiled BitShares version 2.0.170710 on our infrastructure
(seen in Section 6.3). During our experiments, each BitShares instance is member of the witness committee. We fix the interval between block creation to five seconds, the default setting when deploying a private BitShares testnet. The BitShares configuration script we used is published on our GitHub organization.

**Waves.** Similar to BitShares, the Waves platform also allows users to create custom assets and issuing them to others [35]. The adopted consensus mechanism is Waves-NG, a protocol that combines Proof-of-Stake and Bitcoin-NG [12]. Although the blockchain fabric in Waves is decentralized, order matchmaking in Waves proceeds centralized and traders submit new orders to a single matchmaking instance. The submitted order can then only be matched with other orders this matcher knows about. When two orders are matched by a matcher in Waves, the resulting assets are exchanged on the blockchain and this trade is recorded as a transaction. The matcher collects the fees for both matched orders.

---

### Table 1

<table>
<thead>
<tr>
<th>Component</th>
<th>LOC</th>
<th>Line Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trading Mechanism (XChange)</td>
<td>3091</td>
<td>98%</td>
</tr>
<tr>
<td>Overlay Network Logic</td>
<td>947</td>
<td>96%</td>
</tr>
<tr>
<td>Packet (de)serialization</td>
<td>223</td>
<td>100%</td>
</tr>
<tr>
<td>Database Management</td>
<td>112</td>
<td>98%</td>
</tr>
<tr>
<td>Matching Engine</td>
<td>122</td>
<td>100%</td>
</tr>
<tr>
<td>Blockchain Fabric (TrustChain)</td>
<td>633</td>
<td>87%</td>
</tr>
<tr>
<td>Overlay Network Logic</td>
<td>225</td>
<td>90%</td>
</tr>
<tr>
<td>Packet (de)serialization</td>
<td>103</td>
<td>100%</td>
</tr>
<tr>
<td>Database Management</td>
<td>69</td>
<td>92%</td>
</tr>
<tr>
<td>Block Definition &amp; Validation</td>
<td>187</td>
<td>78%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>3724</strong></td>
<td><strong>96.1%</strong></td>
</tr>
</tbody>
</table>

Table 1. Code and test coverage statistics of the XChange and TrustChain implementations. For each (sub)component of our trading mechanism and the blockchain fabric, we show the number of lines of code and percentage of lines covered by automated tests.
We use our commodity cluster to quantify the performance (scalability) of XChange, Waves and BitShares. The detailed specifications of the hardware and runtime environment can be found online\(^6\). Our infrastructure allows us to reserve computing nodes and deploy instances of XChange, BitShares or Waves on each node. We use our experiment framework to quickly deploy applications onto computing nodes, and to automatically extract results from experiments\(^5\). Each experiment is defined by a scenario file, a list of actions which are executed by all or a subset of running instances, at specific points in time after the experiment starts.

## 6.3 Hardware Specifications

We use our commodity cluster to quantify the performance (scalability) of XChange, Waves and BitShares. The detailed specifications of the hardware and runtime environment can be found online\(^7\). Our infrastructure allows us to reserve computing nodes and deploy instances of XChange, BitShares or Waves on each node. We use our experiment framework to quickly deploy applications onto computing nodes, and to automatically extract results from experiments\(^8\). Each experiment is defined by a scenario file, a list of actions which are executed by all or a subset of running instances, at specific points in time after the experiment starts.

## 6.3 Hardware Specifications

We use our commodity cluster to quantify the performance (scalability) of XChange, Waves and BitShares. The detailed specifications of the hardware and runtime environment can be found online\(^7\). Our infrastructure allows us to reserve computing nodes and deploy instances of XChange, BitShares or Waves on each node. We use our experiment framework to quickly deploy applications onto computing nodes, and to automatically extract results from experiments\(^8\). Each experiment is defined by a scenario file, a list of actions which are executed by all or a subset of running instances, at specific points in time after the experiment starts.

## 7 Mechanism Performance Evaluation

We now present our experiments to evaluate the performance of our mechanism. First, we validate the behaviour of XChange with a real-world stock trading dataset. Next, we conduct experiments to answer the following questions: (1) How does the scalability of XChange compare to that of BitShares and Waves? (2) How long does it take to detect double spend fraud in TrustChain? (3) How fault-tolerant is XChange compared to BitShares and Waves?

## 7.1 Validation

We are interested in how our trading mechanism behaves under the burst of activity provided by real-world order book data. It also helps us to verify the correctness of our implementation under realistic order arrivals. We deploy 100 XChange instances and run at most 16 instances on each computing node of our cluster. Every XChange instance acts as matchmaker and connects to ten other random matchmakers in the overlay network when the experiment starts.

We use the LOBSTER datasets discussed in Section 6.1. Each trader accepts the first incoming match notification from a matchmaker. The results of this experiment are visible in Figure 6, where we present the cumulative number of trades completed, since the start of the experiment. The horizontal axis shows the time into the experiment and the vertical axis indicates the cumulative number of trades that have been completed. The experiment lasts for five minutes. We vary the price level update range (complexity) and each experiment with a different complexity, is executed once. As a first observation, note that all runs exhibit roughly the same trading pattern, with a burst in the trading activity around 240 seconds. We observed that this burst correlates with a significant higher CPU usage for specific instances. The CPU utilization of some instances was around 100% for some seconds during that time (the XChange implementation is single-threaded). Note that the dataset with price update range 1 has the lowest number of completed trades at the end of the experiment, compared to the runs with higher order book complexity. It is interesting to note that a more complex order book does not necessarily correlate with a higher number of trades being made during the experiment. Moreover, we suspect that a small difference in order matching throughout different runs, propagates and amplifies. During the experiment, we monitored the XChange instances and searched for indicators that suggest incorrect behaviour, like orders that are matched incorrectly or multiple times.

From Figure 6 and manual verification, we conclude that our mechanism is correctly implemented and handles real-world datasets.

## 7.2 Scalability

We conduct experiments around scalability and perform a stress test with XChange, BitShares and Waves. This work is the first to present a systematic scalability comparison between decentralized blockchain-powered exchanges, to the best of our knowledge. For each run with XChange, BitShares and Waves, we increase the network size (the total number of instances running) with 50, starting with 50 instances in total. We try to deploy as many instances of each platform as possible. For each run with BitShares or Waves with \(n\) instances, we initiate the experiment by starting a single BitShares/Waves instance, which issues two new types of digital assets. The first instance transfers sufficient assets to all other instances, so they can create new orders. After five seconds, we start the other \(n - 1\) instances which then connect to the first instance. When each instance has sufficient funds to create numerous orders, order creation starts. We use the synthetic workload described in Section 6.1 and subject each of the implementations to exactly the same trading workload. Instances create orders during a period of five minutes. Each run during this experiment is executed once.

We define the maximum transaction throughput observed during each run as our scalability metric. Since there is no standardized way to compare throughput between the tested systems, we first have to define what we consider as a transaction. For BitShares, we consider operations as transactions, excluding ones that transfer assets to other instances when the experiment starts (each transaction on the BitShares blockchain consists of one or more operations). For Waves, we consider transactions on the blockchain that are the result of two orders matched by a matchmaker. For XChange, we consider (dual-signed) blocks as transactions and determine the second during which the most of such blocks are created.

---

\(^6\)Omitted for double-blind review.
\(^7\)Omitted for double-blind review
\(^8\)Omitted for double-blind review
Waves instances on each computing node, exceeds the maximum (discussed in Section 4.5). Though we focus on a particular kind of attempts, like modification of blocks (although this is significantly knowledge of blocks in two distinct chains in order to be revealed most challenging kind of fraud. We strongly believe that even at of fraud in TrustChain: the double spend attack, which requires exactly one double spend attack. This user performs this kind of resources), a double spend attack can be detected within minutes of fraud with a probability of 20% when transacting with others. We measure the interval between initiation of the double spend and detection of it by at least one user. As soon as a double spend is detected, its evidence can further be disseminated into the network.

The results are given in Figure 8. The horizontal axis denotes the network size (total number of deployed XChange instances), up to 1,000. The vertical axis shows the time interval between performing the double spend attack and detection of it. We vary the rate at which users request chain fragments from each other. We consider request frequencies of 5, 10 and 25 (where 5 means that the user will request a chain fragment of five unique users every second). For each combination of network size and chain exploration rate, we run the experiment 20 times. The errors bars show one standard deviation of uncertainty. Figure 8 suggests that increasing the chain exploration rate increases the speed of fraud detection, at the expense of increased system resources usage (bandwidth and CPU utilization). Also, observe how variation of the fraud detection speed is higher when exploring chains at a lower rate. Interesting is that effectiveness of fraud detection shows comparable results when the network size increases. Our reasoning for this is as follows: although the specific double spend is more “hidden” in the network, there is also more ongoing effort to detect it.

From this experiment, we conclude that continuous exploration of the distributed ledger is an effective measure to detect even the most challenging kind of fraud. We strongly believe that even at exploration rate of one request per second (which uses minimal resources), a double spend attack can be detected within minutes and handled in a manner as defined by application-specific logic.

During this experiment, we run at most 30 instances on each computing node (we noticed that the peak CPU utilization during this experiment is ≈50% lower compared to the experiment performed in Section 7.2). The global rate at which transactions are made is fixed to 100 transactions per second, an order of magnitude higher than the maximum theoretical throughput of Bitcoin (seven transactions per second). Users initiate transactions in a round-robin fashion with other random users. During the experiment, each user explores the chains of others and attempts to detect fraud. At the start of each experiment, we selected one user to perform exactly one double spend attack. This user performs this kind of fraud with a probability of 20% when transacting with others. We measure the interval between initiation of the double spend and detection of it by at least one user. As soon as a double spend is detected, its evidence can further be disseminated into the network.

The results are given in Figure 8. The horizontal axis denotes the network size (total number of deployed XChange instances), up to 1,000. The vertical axis shows the time interval between performing the double spend attack and detection of it. We vary the rate at which users request chain fragments from each other. We consider request frequencies of 5, 10 and 25 (where 5 means that the user will request a chain fragment of five unique users every second). For each combination of network size and chain exploration rate, we run the experiment 20 times. The errors bars show one standard deviation of uncertainty. Figure 8 suggests that increasing the chain exploration rate increases the speed of fraud detection, at the expense of increased system resources usage (bandwidth and CPU utilization). Also, observe how variation of the fraud detection speed is higher when exploring chains at a lower rate. Interesting is that effectiveness of fraud detection shows comparable results when the network size increases. Our reasoning for this is as follows: although the specific double spend is more “hidden” in the network, there is also more ongoing effort to detect it.

From this experiment, we conclude that continuous exploration of the distributed ledger is an effective measure to detect even the most challenging kind of fraud. We strongly believe that even at exploration rate of one request per second (which uses minimal resources), a double spend attack can be detected within minutes and handled in a manner as defined by application-specific logic.

7.3 Fraud Detection

We now evaluate the efficiency of fraud detection in TrustChain. In particular, we are interested in the time between committing fraud and initial detection of it. Our focus is on the most challenging kind of fraud in TrustChain: the double spend attack, which requires knowledge of blocks in two distinct chains in order to be revealed (discussed in Section 4.5). Though we focus on a particular kind of fraud here, the fraud detection mechanism can detect other fraud attempts, like modification of blocks (although this is significantly easier to detect than a double spend attack).
7.4 Fault Tolerance

Finally, we explore how crashing running instances impact the throughput of subjected systems. We use a similar setup and workload as our scalability experiment in Section 7.2. However, we fix the number of instances to 100 (Figure 7 suggest that all compared systems should be able to reliably handle this network size). During a period of ten minutes, we crash a fraction of all running instances. The time at which an instance crashes, is uniformly distributed during the experiment duration. After ten minutes, we visualize the alteration in throughput during the run. For BitShares and Waves, we determine throughput by counting all transactions/operations in a newly created block, and dividing it by the time in seconds since creation of the last block. For XChange, we compute how many (dual-signed) blocks were created during every interval of ten minutes. We suspect that crashing instances will have a larger impact on the throughput of Waves than on the throughput of other systems, due to centralization of matching activities. We consider failing 10% and 50% of all running instances during the experiment.

We present the result of our fault-tolerance experiment in Figure 9. The horizontal axis shows the time into the experiment in seconds, and the vertical axis shows the throughput. We show the results for our three subjected systems. At $t = 0$, instances start to submit new orders. From Figure 9a, we observe that throughput slowly decreases over a period of ten minutes (we crash one instance every minute). The throughput of BitShares shows a near-linear and predictable decrease in throughput. On the other hand, the throughput of Waves shows a high variation over time and does not decrease in a predictable manner, although it appears to become less variable after five minutes in the experiment.

Figure 9b shows how throughput decreases when half of all running instances fail. XChange roughly shows a linear decrease in throughput as more instances crash. Decrease in throughput of BitShares is very predictable throughout the experiment. Interestingly is that Waves shows a similar pattern as in Figure 9a, although its throughput goes to near-zero after nine minutes.

Based on Figure 9, we conclude that our mechanism is resilient against crash failures. We also conclude that the throughput of Waves is unstable under failures. It is likely that the centralized way of matching orders in Waves contributes to this.

8 Security Analysis

We now analyse the security of XChange. We identify attacks and show how they can be addressed, either by the XChange architecture itself or by trading application built on top of it.

Sybil Attack. The most challenging attack in decentralized networks is the Sybil Attack. A Sybil Attack happens when malicious users create numerous fake identities (Sybils) and interact with them, aiming to subvert the network. In the context of XChange, traders can build an artificial trade history with Sybils. The transactions accompanying these trades might indicate that a large amount of assets have been traded, although in reality nothing is sent or received. Despite numerous proposals, the Sybil Attack remains largely unsolved within decentralized networks [19].

An effective solution for the Sybil Attack, is the use of strong long-lived identities, where each trader can connect their digital identity to a real-world identity. We propose to introduce a trusted identity provider, for instance, a government. Note that this does not violate our design goals (full decentralization) since this trusted intermediary is not responsible for directly facilitating trade. An attestation for one’s identity can then be represented as a transaction on TrustChain, and verified by others. This makes it impractical to quickly generate trusted identities.

Counterparty Fraud. Counterparty fraud happens when a trading partner acts dishonest during a trade, for instance, by not sending assets to another trader. Note that counterparty fraud is detectable by exploring transactions on TrustChain. In particular, absence of a dual-signed proof-of-delivery indicates that one of the parties not sent assets or received them (yet). To further improve security, a trade that transfers a large amount of assets, can be split up into multiple, smaller asset transfers. This also requires construction of a larger number of proof-of-deliveries during a single trade. By doing this, traders exchange assets in an incremental manner, instead of making single, large transfers. A trade can be aborted if one of the parties refuses to transfer his assets to the counterparty.

This decreases value at stake, but prolongs trade.
Reputation mechanisms help traders to make more informed decisions who to trade with, and reduces the risk for counterparty fraud. Trustworthiness scores can either be determined by a trusted third party, or computed by individual traders themselves. We propose to use public TrustChain transactions as input for trust estimation. Prior work devises Sybil-resistant reputation mechanisms, based on TrustChain transactions [34].

Transaction Flooding. Transaction flooding is an attack where an adversary creates and spreads many useless or invalid transactions in the network. In the XChange mechanism, an attacker is able to initiate many orders that will not be fulfilled, consequentially filling the order books of matchmakers and slowing down the matching process. There are several ways to address this attack. First, deploying a reputation mechanism that takes into consideration whether a trader initiated an unusual high number of (useless or invalid) transactions, can reduce impact of this attack (similar to reputation scores of email servers). We should note however that there are situations where creating orders in bulk is not uncommon, for instance, during high-frequency trading. An additional defence can be found in Proof-of-Work mechanisms, where a trader has to perform a small amount of work prior to creating an order [2]. It is then computationally infeasible to flood the network in a short period of time. On the other hand, it increases latency before orders are confirmed, which is what we specifically tried to avoid.

Malicious Witnesses. Witnesses are essential to verify the validity of new orders. However, malicious witnesses can intentionally delay the verification of valid orders or not verify them at all. To counter this, we propose m-out-of-n orders, where new orders are sent to at least n witnesses. The order is only deemed valid by other traders when the order creator can prove that at least m witness nodes signed for the validity of the order (our current implementation assumes m = n = 1). Recall that we extended the TrustChain ledger to support scalable multi-party agreements (see Figure 4e). Depending on the exact values of m and n, and on the selection policy of witnesses, it now becomes more challenging for a single witness to significantly influence order creation.

9 Related Work
We now explore research on decentralized markets. Both work in academia and industry is considered.

Considerable effort has been spent on exploring the performance of decentralized markets and auctions. Fouotura et al. present a peer-to-peer auction based on Law-governed interaction [14]. Peer-Mart is a decentralized auction that enabled trading of peer-to-peer services [18]. Despotovic et al. propose an auctioning mechanism with fast convergence towards efficient trading [11]. Ogston et al. also examine a decentralized auction mechanism where agents build clusters and select a leader for each cluster [33].

Overall, the disadvantages of blockchain-based decentralized markets have not been clearly researched. This work highlights some of the weaknesses of using blockchain technology to facilitate trade. The inception and success of the Bitcoin cryptocurrency interested researchers in the potential of blockchain technology to build decentralized markets, both from a technological and business perspective [30]. Soska et al. propose Beaver, a decentralized and anonymous marketplace with a secure reputation mechanism [38]. Sikorski et al. show how blockchain technology can be used to facilitate a machine-to-machine electricity market [37]. Lee elaborates how blockchain technology can be used to exchange assets in a secure way and improve traditional stock trading [23]. Malinova et al. present a market design for trading with blockchain technology [24]. Their work shows that the counterparty transparency provided by the public ledger yields the highest investor welfare.

Outside academia, numerous applications have been deployed that allow users to exchange goods in a decentralized manner. Swarm City is a decentralized marketplace built on top of the Ethereum blockchain and is designed to operate within the sharing economy [9]. BitShares and Waves are two market that allow users to define and issue their own assets, for a fee [36][35] (see Section 7 for a detailed scalability and fault tolerance evaluation of them). Regarding functionality, Ardor is similar to BitShares and Waves [17]. It separates between a main chain, which only stores transfers of the native ARDR token, and child chains, which record exchange of user-defined transactional tokens.

Another class of decentralized markets improve scalability by removing critical operations from the blockchain. The IDEX and NEX exchanges introduce centralized components by deploying a server to store all orders and perform order matching, to ensure continuous trade [21][13]. OasisDex allows trading of a select number of tokens and stores orders on the chain, however, it lacks a matching engine [31]. The 0x project defines a hybrid protocol to facilitate decentralized trading of Ethereum tokens by maintaining the order book outside the blockchain [43]. Loopring is an open protocol for building decentralized markets, by recording order completion on the blockchain and relying order information outside the blockchain [15]. All these applications and protocols rely on establishing global consensus to keep the blockchain secure.

Finally, we show decentralized marketplaces that are not powered by with blockchain technology. Bisq is a decentralized marketplace for cryptocurrencies and offers built-in privacy by routing messages over the Tor network (which significantly increasing latency) [4]. The OpenBazaar platform provides an alternative for established e-commerce platforms like Amazon and Alibaba [32].

10 Conclusions
We presented XChange, a blockchain-based mechanism to trade at scale. Unlike most related decentralized markets built with blockchain technology, our trading mechanism does not require global consensus. We build upon a blockchain fabric, TrustChain, to store all orders and transactions. In addition, scalability of TrustChain has been improved by adding support for concurrent block creation. Our mechanism is specifically built for segmented markets where trader interaction is clustered. By allowing developers to define their own wallets and settlement providers, we facilitate trading beyond cryptocurrencies and digital tokens.

We evaluated the performance of XChange, while comparing our work with state-of-the-art, blockchain-based marketplaces. After verifying the correctness of our implemented on real-world stock trading data, we conclude that our mechanism shows superior scalability compared to BitShares and Waves. Fraud performed with blocks on the distributed ledger can be efficiently detected by network participants within seconds, even for larger networks. Finally, we concluded that our mechanism is fault-tolerant against crash failures of individual instances.