XChange
A Universal Mechanism for Asset Exchange between Permissioned Blockchains

de Vos, Martijn; Ileri, Can Umut; Pouwelse, Johan

DOI
10.1007/s11280-021-00870-x

Publication date
2021

Document Version
Final published version

Published in
World Wide Web

Citation (APA)
de Vos, M., Ileri, C. U., & Pouwelse, J. (2021). XChange: A Universal Mechanism for Asset Exchange between Permissioned Blockchains. *World Wide Web*, *24*(5), 1691-1728. https://doi.org/10.1007/s11280-021-00870-x

World Wide Web (2021) 24:1691–1728
https://doi.org/10.1007/s11280-021-00870-x

XChange: A Universal Mechanism for Asset Exchange between Permissioned Blockchains

Martijn de Vos[1] · Can Umut Ileri[1] · Johan Pouwelse[1]

Received: 20 May 2020 / Revised: 6 December 2020 / Accepted: 16 February 2021 /
Published online: 13 March 2021

Abstract

Permissioned blockchains are increasingly being used as a solution to record transactions between companies. Several use cases that leverage permissioned blockchains focus on the representation and management of real-world assets. Since the number of incompatible blockchains is quickly growing, there is an increasing need for a universal mechanism to exchange, or trade, digital assets between these isolated platforms. There currently is no *universal* mechanism for inter-blockchain asset exchange without a requirement for trusted authorities that coordinate the trade. We address this shortcoming and present XChange, a universal mechanism for asset exchange between permissioned blockchains. To achieve universality and to avoid trusted authorities that coordinate a trade, XChange does not provide atomic guarantees but leverages risk mitigation strategies to reduce value at stake. Our mechanism records the specifications and progression of each trade within records on a distributed log. XChange reduces the economic gains of adversaries by bounding the total amount of fraud they can commit at any time. After having committed fraud, an adversary is forced to finish its ongoing trades before it can engage in new trades. We first present a four-phased protocol that coordinates an asset exchange between two traders. We then outline how trade records can be stored on TrustChain, which is a lightweight distributed ledger specifically built for the tamper-proof storage of data elements. We implement XChange and conduct experiments. Our experiments demonstrate that XChange is capable of reducing the economic gains of adversaries by more than 99.9% when replaying a real-world trading dataset. A deployment on low-resource devices reveals that the latency added to a trade by XChange is only 493 milliseconds. Finally, our scalability evaluation shows that XChange achieves over 1’000 trades per second and that its throughput, in terms of trades per second, scales linearly with the system load.

Keywords Blockchain interoperability · Universal asset exchange · Blockchain asset trading · Permissioned blockchains

This article belongs to the Topical Collection: *Special Issue on Emerging Blockchain Applications and Technology*
Guest Editors: Rui Zhang, C. Mohan, and Ermyas Abebe

✉ Martijn de Vos
m.a.devos-1@tudelft.nl

Extended author information available on the last page of the article.

1 Introduction

Bitcoin, introduced in 2008, has revolutionized the field of digital currencies by demonstrating that it is possible to devise a secure cash system without a bank [38]. The goal of Bitcoin is to realize a payment system through the secure management of a native currency on a distributed ledger. The creation of this currency is controlled by miners participating in a voluntary process known as mining. The collective efforts of miners ensure the security of Bitcoin and prevent illegitimate coin creation. Miners invest computational power to include valid transactions on the blockchain, which is a tamper-proof distributed ledger that consists of blocks. One of the compelling features of a blockchain is the ability to securely record and validate user-issued transactions without trusted authorities, even in the presence of mutual distrust between participants.

Participation in many deployed blockchains is open for everyone and does not require the explicit approval from authorities unlike traditional banking systems. Even though blockchain technology provides the means to maintain a distributed ledger without trusted authorities, open enrollment is not required for many industrial use cases, or is even undesirable. For instance, when two companies leverage blockchain technology to securely record their transactions, read and write access to the distributed ledger is most likely limited to a few selected employees or operators. Over the past few years, there has been a sharp increase in the development and deployment of private, or *permissioned* blockchains [1, 10, 55]. In contrast to a public blockchain like Bitcoin, membership in a permissioned blockchain is managed by an authority that approves the participation of each peer. The identity under which a peer operates is linked to a real-world persona, which reduces the likelihood of Byzantine behavior and network threats like the Sybil Attack [16]. Permissioned blockchains usually adopt a classical consensus model designed for networks with static membership, e.g., Practical Byzantine Fault Tolerance (PBFT) [8]. Considerable efforts in permissioned blockchains have been made by projects such as Hyperledger Fabric [1], R3 Corda [6], Quorum [37] and BigchainDB [35]. Permissioned blockchains have the potential to increase the efficiency of traditional business processes in industries like logistics, energy management and trade supply chains [55].

Several use cases that record transactions on a permissioned blockchain revolve around the representation and management of real-world assets on a distributed ledger [26]. Advancements in blockchain technology have resulted in numerous platforms on which companies can issue and manage digital assets. There currently is a proliferation of different types of assets, fragmented across many different blockchain implementations [5]. In public blockchains, almost 200 000 different assets are being managed on the Ethereum blockchain only.[1] A recent Forbes report reveals that at least 50 major companies, each valued at least at $1 billion, are exploring blockchain technology for asset management and trading [13]. As industry's adoption of blockchain technology is increasing, a similar asset proliferation will occur with permissioned blockchains. Unfortunately, there is no *universal* mechanism to exchange (trade) assets between isolated distributed ledgers without the involvement of a trusted third party. Research and developments in distributed ledger technology mostly focus on the deployment of new domain-specific blockchains, whereas interoperability issues are mostly ignored [47, 58]. In particular, there is a lack of research

[1] See https://etherscan.io/tokens

on the interoperability of permissioned, industry-grade blockchains [39, 54]. Interoperability concerns are particularly relevant when leveraging distributed ledgers for trading, as a single trade consignment can involve various isolated blockchains [20]. Given the inevitable growth of permissioned blockchain platforms, we argue that a universal mechanism for asset exchange between these platforms is a growing necessity.

We present *XChange*, a universal mechanism for asset exchange, or *trade*, between permissioned blockchains.[2] XChange coordinates trade between separate permissioned blockchains by storing trade records in a distributed log, also see Figure 1. Our solution is independent of the technical characteristics of the involved blockchains and does not require modifications to blockchain applications that are already operational. An asset exchange in XChange proceeds through a sequence of alternating, unilateral asset transfer operations (payments) between two parties. This is comparable to how many electronic markets (e.g., eBay) operate, where a party only initiates a payment back to the counterparty after having received a payment first. Sequential payments, however, introduce a risk of losing economic value to the other party, since the other party is now able to "steal" assets during a trade [30]. This fraud is called *counterparty fraud* and it is a severe concern in many electronic marketplaces that facilitate peer-to-peer trading [43]. For this reason, we argue that any asset trading mechanism must either prevent counterparty fraud or punish a participant that has committed this fraud upon its detection.

To address counterparty fraud, blockchain-based asset exchange often provides *atomic* guarantees. Atomicity in this context implies that a trade either exchanges all assets between involved parties or exchanges nothing. We find that the security of existing trade solutions either (1) relies on (semi-trusted) authorities to ensure that assets are securely exchanged, or (2) relies on the availability of specialized transactions by the blockchains that manage the assets being traded. Relying on authorities is the standard approach when trading assets managed by public blockchains, e.g., by using the services of a cryptocurrency exchange. In a permissioned setting, however, this approach requires the participation of these intermediaries in the involved blockchains, which is not always allowed by their network operators. Asset exchange mechanisms that depend on specialized transactions, e.g., atomic swaps [40], are not universal enough to support asset exchange between any pair of permissioned blockchains.

In contrast to existing solutions, XChange particularly focuses on the *detection* of counterparty fraud. We argue that the detection of counterparty fraud during a trade is sufficient, since misbehavior can always be traced back to a real-world identity, and optionally be punished by an external authority. To detect counterparty fraud, XChange requires traders to append tamper-proof trade records to a distributed log. By recording the initiation of each trade, conducted payments, and the completion of a trade, participants can detect if a malicious trader has committed fraud and then refrain from trading with that party.

XChange does not provide atomic trade guarantees; however, it bounds the economic gains of adversarial parties by introducing two risk mitigation strategies. First, XChange allows a trade to gradually complete through multiple, smaller payments. We refer to this technique as *incremental settlement*. With incremental settlement, traders themselves decide how much risk they are willing to take, and specify how much economic value they put at stake. Our second risk mitigation strategy is to bound the value that traders are entrusted with during ongoing trades. This bound is decided by traders themselves and enables a

[2] We use the terms "exchange" and "trade" interchangeably in this work.

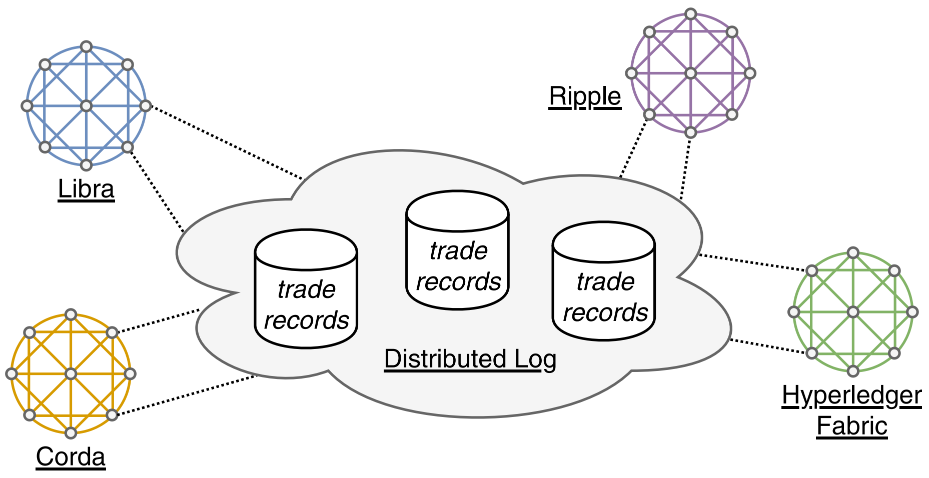


Figure 1 XChange coordinates the asset exchange between permissioned blockchains by storing trade records in a distributed log. This enables traders to detect if a party has committed fraud during an ongoing trade

trader to still be engaged in multiple lower-risk trades. XChange forces an adversarial party to finish its ongoing trades first before it can engage in other high-valued trades. We prove that this approach bounds the economic gains of adversaries. Since XChange assumes static membership through well-defined identities, it prevents a situation where a participant that has committed counterparty fraud can re-joins the network under a new digital identity and commit fraud again (the whitewashing attack [19]).

In this work, we first present and classify existing mechanisms for cross-blockchain asset exchange. We then outline our solution and describe the XChange protocol. We deploy XChange using a tamper-proof, distributed log with low overhead, a technology that predates Bitcoin [22]. Specifically, we leverage an existing solution, TrustChain, that is built for the secure logging and accounting of generic data elements [41]. Our experiments with real-world trading data reveal that our risk mitigation strategies can reduce fraud gains by 99.9%. By conducting a trade between two Raspberry Pis, we quantify that the added latency by XChange is only 493 milliseconds. Additional experiments on our compute cluster reveal that XChange can handle over 1’000 trades per second and that its throughput scales linearly with the system load.

The main contribution of this work is four-fold:

1. The XChange *trading protocol* which specifies how assets are exchanged between permissioned blockchains by storing trade records in a distributed log (Section 5).
2. *Risk mitigation strategies* that lower the risk for traders and bound the economic gains of adversaries committing counterparty fraud.
3. Improvements of TrustChain, a tamper-proof, distributed log used by XChange. Our improvements enables concurrent transactions and increase scalability (Section 7).
4. A functional, open source *implementation* of the XChange trading protocol (Section 8.1).
5. *Experimentation* around the security, resource usage and scalability of XChange, conducted on multiple low-resource devices and our compute cluster (Sections 8.2–8.4).

2 Related work and problem description

Achieving interoperability between blockchains is a challenging problem and remains largely unsolved [7, 54, 59]. Most research in this direction considers cross-chain interactions between permissionless blockchains [30]. There is little research on how to achieve interoperability between permissioned blockchains, even though this is also a concern in private environments. We first discuss existing solutions that address asset exchange between different blockchains, ranging from approaches that rely on a central authority to trust-less trading mechanisms using specialized transactions or intermediate blockchains. Based on our findings, we then formulate the requirements for our asset exchange mechanism.

2.1 Central authorities

A common approach to exchange blockchain-based assets is by using the services of a central authority. A trade using a central authority completes as follows: two parties that agree on a trade transfer the assets for sale to one of the wallets owned by the authority. When this intermediary has received both assets, it finishes the exchange by transferring the appropriate assets to the other party. In this approach, the authority holds (temporary) ownership of the assets to be traded. Relying on a central authority removes counterparty risk for the trading parties, but it requires both parties to have faith that the intermediary does not default or compromise their assets.

Trade through a central authority can facilitate value exchange between an extensive range of different blockchains, as long as the intermediary maintains wallets on the involved blockchains and can issue transactions in these systems to transfer the assets. This is usually not an issue in permissionless blockchains since anyone can create accounts or wallets by generating a new cryptographic key pair. Centralized cryptocurrency exchanges often facilitate asset trading across numerous permissionless blockchains. Some cryptocurrency exchanges process transactions worth millions of dollars in total daily.[3] In a permissioned blockchain environment, however, a central authority coordinating an asset exchange requires explicit approval from the operator to read and write transactions on the involved distributed ledgers. Allowing new parties in a permissioned blockchain might be undesirable by operators since it introduces additional legal and operational risks.

There have been various efforts to mitigate the trust issues surrounding centralized exchanges and trusted authorities while still maintaining a centralized infrastructure. TEX is a centralized exchange that uses an off-chain settlement solution for trust-less asset trade [29]. TEX is resilient against the front-running attack where insider information is exploited to gain a financial advantage while trading. Tesseract leverages trusted hardware, e.g., Intel SGX, to build a secure cryptocurrency exchange that also addresses the front-running attack [4]. The Arwen trading protocol is another protocol to securely trade cryptocurrencies through a centralized exchange without giving up ownership of the assets to the exchange [24].

2.2 Atomic swaps

The *atomic swap* is a protocol that is commonly used to exchange assets between different blockchains, without need for a central authority [25]. Atomic swaps enable two parties to

[3]See https://coinmarketcap.com/rankings/exchanges

exchange blockchain-based assets in an atomic manner: the asset exchange either completes or fails for both parties at any given time.[4] Atomic swaps eliminate the risk of losing assets to an adversarial trader during the exchange. The main idea is that trading users lock their assets in a specialized transaction on the blockchain in such a way that no single party can claim both locked assets. This is achieved with *Hash-Timelock Contracts* (HTLCs), a special transaction that leverages hash locks and time locks. A hash lock is a restriction that prevents the transfer of assets until the pre-image of a provided hash is revealed. A time lock is a primitive that locks assets until a specific time. They prevent the assets being traded from being locked up indefinitely during an atomic swap. This time lock should be well above the block confirmation time of the underlying blockchain to prevent the loss of assets during a blockchain reorganization. In practice, the duration of the time lock is often fixed to several hours.

We further explain the atomic swap by considering a trade with Bitcoin and Ether (the native token of the Ethereum blockchain). Figure 2 visualizes an atomic swap between two parties, Alice and Bob, where Alice sells her Bitcoin in return for Ether. The basic atomic swap, described by Tier Nolan [40], consists of the following six steps:

Step 1. Alice generates a secret value s and computes $H(s)$, where $H(\cdot)$ is a secure hash function.

Step 2. Alice submits a hash-timelock transaction T_1 to the Bitcoin blockchain, locking her Bitcoin and using $H(s)$ for the hash lock. A party can claim the Bitcoin held by T_1 with another transaction that provides s, within a specific time duration.

Step 3. Alice sends $H(s)$ to Bob using any communication medium.

Step 4. Bob submits a hash-timelock transaction T_2 to the Ethereum blockchain, locking his Bitcoin and also using $H(s)$ for the hash lock.

Step 5. Alice claims the Bobs' Ether locked in T_2 by submitting a transaction, T_3, to the Ethereum blockchain, containing s. T_3 unlocks the hash-lock in T_2. This reveals pre-image s to Bob.

Step 6. Bob now claims Alice's Bitcoin locked in T_1 by submitting a transaction, T_4, to the Bitcoin blockchain, containing s. The asset exchange is now complete.

The above protocol requires a total of four transactions. Note how Alice is not able to claim her assets without providing the opportunity for Bob to claim his assets.

Atomic swaps enable asset exchange between a wide range of blockchains. Even though they are an interesting proposition for cross-chain asset trade, we describe three deficiencies of this technique. First, atomic swaps can only be used when trading assets between distributed ledgers with support for specific programming constructs, such as time-locked and hash-locked transactions. Both blockchains are also required to support the same hashing algorithm. Second, atomic swaps require traders to lock their assets using a hash-timelock transaction. This enables a Denial-of-Service attack where a party can intentionally retain the assets of a counterparty, denying the counterparty from using the locked assets for other purposes. Third, atomic swaps can be unfair for one of the parties since the swap initiator has a time window after both parties have locked their assets, during which it can decide to abort the swap [23]. This window enables price speculation by the swap initiator by keeping the assets of the other party locked until the asset price goes in the favor of the initiator.

[4]We remark that the atomicity of the atomic swap protocol depends on the security of the underlying blockchains. If one of the blockchains is compromised by adversaries, atomicity during asset exchange cannot be guaranteed and one of the parties can lose its funds to the counterparty.

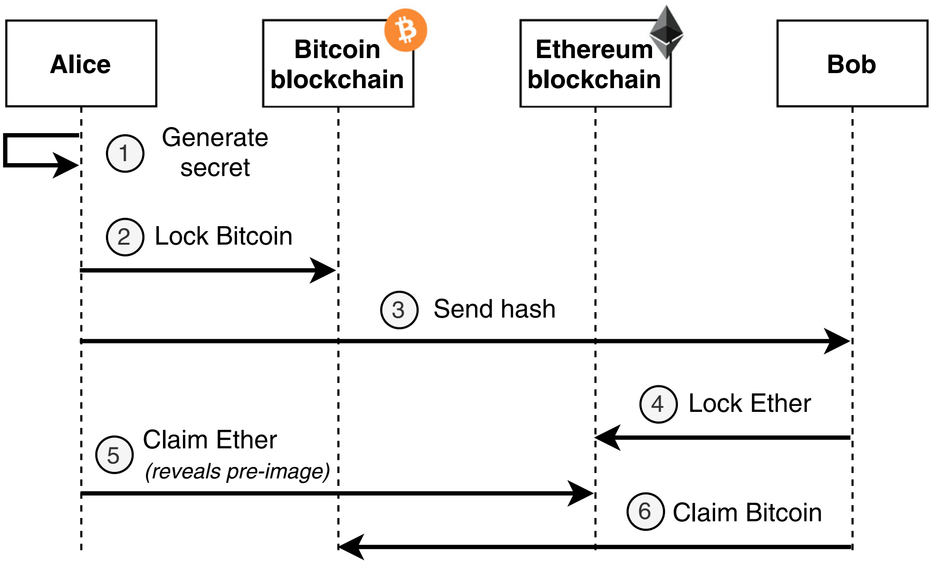


Figure 2 Sequence diagram of a successful HTLC-based atomic swap between Alice and Bob.

2.3 Notary schemes

Notary schemes are another solution for asset exchange where the approval by a group of credible nodes (notaries) is required to perform some operation. Notary schemes aim to partially alleviate the trust issues arising when relying on a central authority through the approval by a group of semi-trusted notaries instead. These notaries reach consensus on the occurrence of particular events, e.g., on the inclusion of a transaction on a distributed ledger. Compared to an asset exchange through a central authority, notary schemes assume a weaker trust model and can often withstand adversarial behavior of a fraction of the notaries.

AgentChain is an asset exchange system that is based on multi-signature schemes [33]. Each user can act as a trading operator, which together form trading groups. Assets are locked in a multi-signature wallet that requires a multi-signature to unlock. Users can choose to trade within a specific trading group, e.g., based on the reputation of the trading group. If a trading group acts malicious, a user can upload evidence to the blockchain upon which all assets managed by that trading group is transferred back to users.

An earlier version of the Interledger protocol, ILPv1, used intermediate notaries (also called *connectors*) to conduct payments across different ledgers [53]. These payments are realized through conditional payments and are coordinated by a different group of connectors for every involved ledger. Interledger uses payment paths where additional intermediate platforms and their connectors are used to exchange assets between ledgers that do not have a direct connection. Only when a particular condition is met, the payment is conducted.

2.4 Blockchain bridges

Another approach to cross-chain trade uses bridging techniques, where an intermediate blockchain mediates asset exchange between different blockchains. Most bridging approaches execute the atomic swap protocol for the exchange process of assets but provide additional primitives and interoperability features for communication between blockchains.

Blocknet is a platform for inter-blockchain routing and facilitates the exchange of cryptocurrencies between blockchains [9]. Blocknet consists of two main components: XBridge and XRouter. XBridge is a decentralized protocol that coordinates atomic swaps between permissioned and unpermissioned blockchains. XRouter provides a peer-to-peer overlay network consisting of clients running the SPV protocol, therefore avoiding the need to download the full blockchain to verify the inclusion of particular transactions. Blocknet secures its transactions through a Proof-of-Stake consensus protocol. Furthermore, Blocknet provides a decentralized exchange where traders can indicate their trade interests through orders. A blockchain connected to Blocknet requires the implementation of time-locked transactions.

ARK is a platform for cross-chain asset exchange that shares similarities with Blocknet [52]. ARK enables users to build custom blockchains (a "BridgeChain") that is powered by the ARK blockchain. To facilitate asset exchange between different blockchains, ARK acts as an intermediate blockchain in the trade process. The latter is achieved through the smart bridge protocol, relying on atomic swaps to exchange value across chains. The ARK blockchain achieves transaction security through a Delegated Proof-of-Stake (dPos) consensus algorithm, where stakeholders vote for a small committee that appends blocks to the ARK blockchain.

The Proof-of-Authority (POA) blockchain is an Ethereum-based permissioned blockchain that provides several tools for interoperability [51]. The POA blockchain is secured by the Proof-of-Authority consensus mechanism, where validating nodes stake their reputation to secure the blockchain. The TokenBridge protocol enables users to not only exchanges assets between Ethereum-based platforms, but also facilitates arbitrary data transfer.

2.5 Sidechains

Sidechains provide the means to exchange assets between blockchains that share similarities, e.g., that run a particular consensus algorithm [3, 48]. In essence, a sidechain is a blockchain that is attached to a parent chain. With a two-way pegged sidechain, assets residing on the parent chain can securely be moved to the sidechain and vice versa. These transfers lock the assets on one chain and re-create them on the connected sidechain or parent chain. A related scheme is federated pegged sidechains [15]. In a federated pegged sidechains, assets moving to another chain are controlled by a group of notaries, making this approach comparable to notary-based solutions (see Section 2.3). Liquid is a deployed sidechain to the Bitcoin blockchain and can be used to quickly trade Bitcoin-derived currencies [15].

2.6 Internet-of-Blockchains

We now describe two solutions that aim to devise an "Internet-of-Blockchains", where a single blockchain controls many sub-chains. The Cosmos project, introduced by the Interchain Foundation, builds a network of heterogeneous blockchains that can seamlessly interact with each other [32]. The Cosmos Hub is the leading blockchain that connects many other blockchains, called zones. Each zone can have its own governance rules and is secured using the Tendermint BFT consensus protocol. Tokens can quickly be exchanged between the Hub and zones using the Inter-Blockchain Communication (IBC) protocol. To interact with blockchains external to Cosmos, there is a particular zone, called a *bridging zone*. The bridging zone keeps track of transactions and blocks persisted on external blockchains, e.g., Ethereum.

The system architecture of Polkadot, introduced by Gavin Wood, is similar to Cosmos [57]. Polkadot introduces a single relay chain that is responsible for the coordination of one or more parachains. Polkadot secures its chains through a custom consensus algorithm, inspired by Tendermint [31] and HoneyBadger [36]. Compared to Cosmos, Polkadot aims for a more generic message-passing algorithm between parachains that can not only transfer value.

Both Cosmos and Polkadot can facilitate the effortless exchange of assets between zones or parachains. However, they have limited capabilities for interaction with external blockchains. To benefit from the advantages that Cosmos and Polkadot provide, all involved companies must fully commit to the same blockchain platform, which is hard to achieve in practice. Therefore, the advantage of Internet-of-Blockchains is questionable for industrial use cases, and a less demanding approach might be preferred when trading assets.

2.7 The interledger protocol V4 (ILPv4)

The Interledger Protocol V4 (ILPv4) is a protocol for conducting payments between different ledgers [44]. Although the protocol primarily resolves around one-way asset transfers, it can also be used to exchange assets between different ledgers. ILPv4 maintains a peer-to-peer payment network consisting of different connectors that can transfer value across heterogeneous networks within ILP packets. In comparison to ILPv1 (discussed in Section 2.3), ILPv4 is designed around the fast transfer of low-valued payments. ILPv4 drops the requirement for ledger-based payments since they can be slow to complete and can lead to capital retention, similar to atomic swaps. A sender and a connector are assumed to have funds on some shared network, e.g., they can maintain a unidirectional or bidirectional payment channel if an appropriate blockchain is used.

An ILPv4 payment between a sender S and a receiver R using two connectors proceeds as visualized in Figure 3. First, S and R create a shared secret, which will act as the condition for the payment (step $\textcircled{1}$). Then, S will prepare a `prepare` packet that contains the details of the upcoming payment and the details of the agreed-upon condition (step $\textcircled{2}$). This packet is sent to an available connector, which forwards the packet to subsequent connectors until the packet reaches the receiver R. When receiving the `prepare` packet, R determines the validity of the payment as stipulated by a higher-level protocol and can either reject the payment by sending a `reject` packet back, or accept the payment by responding with a

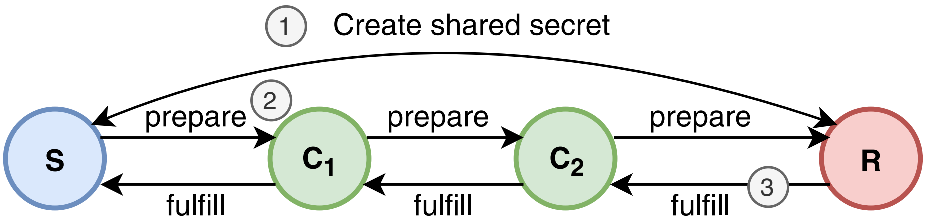


Figure 3 A successful ILPv4 payment from a sender S to a receiver R, using two connectors C_1 and C_2

`fulfill` packet (step ③). The `fulfill` packet contains the pre-image of the agreed-upon condition. Connectors forwarding a `fulfill` packet verify the included pre-image against the payment condition in the previously received `prepare` packet.

The Hyperledger Quilt project provides a Java implementation of the Interledger protocol for permissioned blockchains [27]. The project provides a set of rules for enabling ledger interoperability, formats for network packets and a framework for designing applications that leverage ILPv4.

2.8 Information exchange

We end with a brief discussion on techniques for the exchange of private information across different ledger implementations. Whereas asset exchange involves transfer of ownership, information exchange requires that the buyer does not learn the information without the seller receiving something in return. An information exchange is said to be *fair* when this aforementioned property holds [2].

The FairSwap protocol is the most advanced approach in this direction and ensures a fair exchange of digital goods by leveraging smart contracts and arithmetic circuits [17]. The protocol introduces a *proof-of-misbehaviour* that proves if a seller misbehaves during an exchange. This proof is computationally cheap to construct. The OptiSwap protocol extends FairSwap by incorporating an interactive dispute resolution protocol, reducing the communication overhead of FairSwap [18]. Delgrado et al., describe a protocol for fair data exchange based on the Bitcoin scripting language [14]. The protocol is based on a new primitive, private key-locked transactions, that allow the atomic exchange of a private key for Bitcoin. This private key is then used to decrypt the traded information.

2.9 Comparison and summarization

Table 1 summarizes existing approaches to cross-chain asset trading, and also shows the approach proposed in this work. We assess these approaches based on the following three criteria:

1. **Universal.** does the approach enable asset exchange between any permissioned blockchain?

Table 1 A comparison of approaches to exchange assets between permissioned blockchains

Approach	Universal?	Avoids trusted parties?	Guarantees atomic exchange?
Central Authorities	✓	✗	✗
Atomic Swaps	✗	✓	✓[a]
Notary Schemes	✓	✗	✗
Bridging	✗	✓	✗
Sidechains	✗	✓	✓
Internet-of-Blockchains	✗	✓	✓
Interledger Protocol V4	✓	✗	✗
Information Exchange	✗	✓	✓
XChange (this work)	✓	✓	✗[b]

[a]If the involved parties claim there assets before the time lock expires

[b]But the economic gains of adversaries are limited

2. **Avoids trusted parties.** does the approach require a trusted party to mediate in the trade? We also consider trusted committees or notaries as a trusted party, even though approaches leveraging semi-trusted authorities often assume a weaker trust model.
3. **Guarantees atomic exchange.** does the approach provide an atomic exchange of assets? An atomic exchange guarantees that both parties either exchange all assets, or nothing happens.[5]

Table 1 shows that five out of the eight discussed approaches for asset trading are not universal and cannot facilitate asset exchange between any permissioned blockchain. Asset exchange through a central authority or notaries can support an extensive range of different ledgers but requires the active participation of these authorities in the involved blockchains. The Interledger Protocol is specifically designed for broad adoption and high interoperability between blockchains, but requires semi-trusted connectors to facilitate the payment. We observe that most asset trading mechanisms avoid the need for trusted parties and leverage cryptographic techniques to facilitate trade between different blockchains. Finally, we notice that half of the identified approaches do not guarantee an atomic asset exchange.

2.10 Problem description

Our analysis of existing asset exchange approaches indicates that no solution is universal, avoids trusted parties, and guarantees an atomic exchange. We also observe that there are no solutions that are both universal and avoid trusted parties, to the best of our knowledge. We argue that any mechanism with these two properties requires a compromise on the atomicity criteria. As pointed out by literature on e-commerce, trade atomicity can be addressed by either (1) leveraging specific cryptographic techniques or (2) by using escrow services [45]. Approach (1) violates the universality criteria: it lowers the applicability of our solution since the involved blockchains now require the availability of cryptographic techniques. Approach (2) violates the criteria to avoid trusted parties since an asset exchange is now executed by an escrow.

Even without atomic trade guarantees, we can ensure that the risks of losing funds to the counterparty are manageable. We believe that the Interledger Protocol V4 is the closest to our envisioned universal cross-blockchain value exchange since it makes no assumptions on the technical capabilities of involved payment networks and operates with manageable risks. However, value exchange with the Interledger Protocol does not directly proceed between two traders and is coordinated by intermediate connectors instead. We now formulate three requirements for our asset exchange mechanism:

1. **Universality**. We require that our mechanism enables the exchange of assets between a large range of permissioned blockchains. In particular, asset exchange using our mechanism should not be limited to a selected number of blockchain architectures with specific features or with support for particular transaction types. We argue that this requirement is critical for broad adoption of our mechanism.
2. **Avoid reliance on trusted parties**. We require that our mechanism avoids dependence on trusted parties to settle a trade. Asset exchange should proceed through direct interactions and payments between traders.

[5]In some problem domains, this is also referred to as a *fair* exchange [2].

3. **Manage counterparty fraud**. To achieve universality, we believe that we have to forego the atomicity requirement. Without atomic guarantees, we must address the situation where a trader might actively try to commit fraud for economic gains. Our solution requires adequate measures to manage counterparty fraud during ongoing trades.

These requirements directly lead to the following research question: how can we devise a universal mechanism to exchange assets that are stored on different permissioned blockchains, without having trusted authorities involved in the exchange and with manageable counterparty risk?

3 Solution outline

To avoid the involvement of an intermediary during trades, we leverage an *accounting mechanism* to make all trade activities public and openly accessible to involved traders. Individual accountability is a long-standing and widely used approach in electronic commerce to detect malicious behaviour and to deter fraudsters [12, 28]. By logging full trade specifications, a trader can build a profile of other traders and decide whether it wants to engage in a particular trade, without the involvement of trusted authorities. This approach enables traders to operate according to their own business rules and to manage the economic value at stake.

In this section we outline XChange, our universal mechanism for asset exchange between permissioned blockchains. In XChange, a trade between two traders A and B is modeled as a sequence of payments between the trading parties. At the minimum, a trade involves two payments, one from A to B and one from B to A. W.l.o.g., assume that A initiates the first payment to B in a particular trade. A complication during this trade could arise when B refuses to conduct a payment back to A, after having received a payment from A. In this situation, B has committed *counterparty fraud* since it compromised the assets that A has sent to B. In general, the party that conducts the first payment during a trade is exposed to *counterparty risk* where this party can lose assets to the counterparty without receiving a payment in return.

3.1 Recording trades

We address fraud concerns by storing trade records in a tamper-proof distributed log. This distributed log then enables XChange traders to detect if a party might have committed fraud during an ongoing trade, further discussed in Section 3.2. If so, a trader refrains from starting a trade with the suspected party. We store records of every trade, which makes it difficult for a trader to hide the existence of a specific trade or to unilaterally revert the status of an ongoing trade to a prior state. Each record in the distributed log is digitally signed by its creator and therefore irrefutably created by a specific peer. We envision that the distributed log can also be audited by external authorities to resolve potential disputes that would arise during the trade procedure. However, we consider the details of such audits beyond the scope of this work. The technical requirements of the distributed log are later discussed in Section 4.

Before we show how trade specifications are recorded, we first elaborate on two implications of using a shared log. The first implication is that our solution requires participants to agree on the same distributed log when trading assets using XChange. However, in contrast to Internet-of-Blockchains solutions such as Cosmos and Polkadot, XChange does not require businesses to migrate their deployed ledger applications to a different environment. Instead, businesses can voluntarily leverage our mechanism and join the XChange peer-to-peer network without any changes to existing applications. This approach lowers the adoption barrier of XChange by interested parties. The second implication pertains to privacy concerns, arising from the full accounting of trade specifications. We acknowledge that it might be undesirable to publicly record trade information in specific situations since the records can reveal sensitive business practices. However, since privacy preservation will likely require additional mechanisms and cryptographic techniques, we consider privacy concerns beyond the scope of our work.

We have considered an alternative design where trade records are stored by the ledgers that are involved in the trade. Even though this design would avoid the need for a shared log, it would result in the fragmentation of trade records across potentially many different ledgers, making it infeasible to accurately determine in which trades a specific trader is currently involved. Furthermore, a user can be unable to accurately build profile information of another trader since this user might not have the appropriate credentials to inspect the records and transactions on a specific ledger. This design would also require logic to store XChange trade records within all supported blockchain environments, requiring significant implementation efforts.

We show a part of the distributed log in Figure 4 and highlight four records that together describe a completed trade between two traders, Alice and Bob. This trade exchanges tokens that are managed by a Hyperledger Fabric and a Ripple ledger. The lower part of Figure 4 shows parts of the Hyperledger Fabric and Ripple ledger. A completed trade that has been stored in the distributed log consists of the following three record types:

1. An `Agreement` record contains the specifications of an upcoming trade, e.g., the agreed amount of assets that will be exchanged between the traders. It also includes information on which party conducts the first payment during the upcoming trade. The `Agreement` record bears the digital signature of both traders and can be appended to

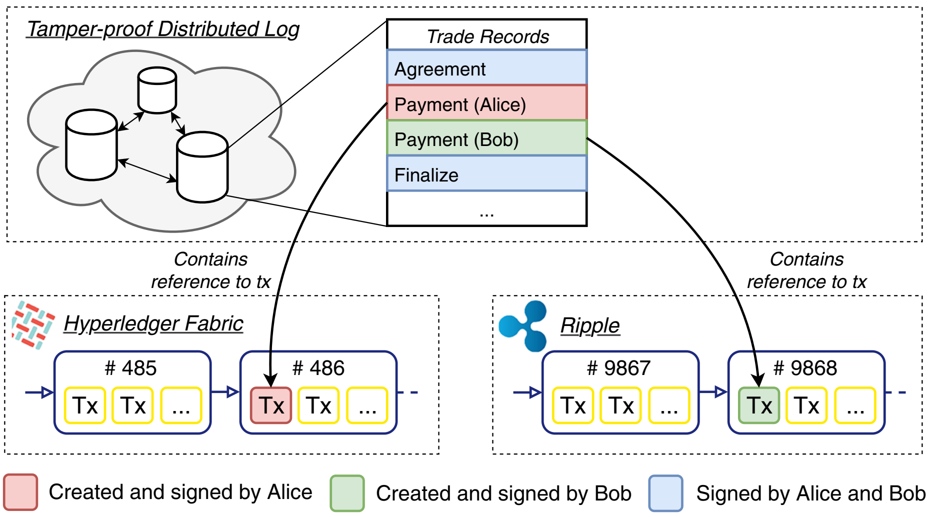


Figure 4 High-level overview of our XChange trading mechanism. In this example, Alice sells some FabTokens that are managed by Hyperledger Fabric to Bob, who pays Alice in XRP (Ripple) tokens. Full trade specifications are stored in a distributed log

the distributed log by any of the traders. We further describe this record type, and the other two record types below, in our protocol description (see Section 5).

2. A `Payment` record contains the details of a specific payment that has been conducted during a trade. This record includes the identifier of the newly issued transaction that transfers assets in the involved blockchain network. For instance, the `Payment` record created by Alice in Figure 4 contains a reference to the transaction that she submitted in the Hyperledger Fabric network. Likewise, the `Payment` record created by Bob points to his transaction in the Ripple network. By including the identifier of the transaction in this record, the trading counterparty, and other traders, can verify if the payer transferred the assets. Others can verify the validity and inclusion of the transaction reference by the `Payment` record by inspecting the appropriate blockchain.
3. A `Finalize` record completes a trade. A `Finalize` record is appended to the distributed log by the party that received the last payment during the completed trade.

In addition to these three record types, XChange also includes the `Order`, `CancelOrder`, and `CancelTrade` records. The `Order` and `CancelOrder` records are used when creating a new order and when canceling an unfulfilled order, respectively. These two record types are further discussed in Section 5. The `CancelTrade` record can be appended to the distributed log to unilaterally abort the trade if one of the parties becomes inactive during a trade. This feature is later discussed in Section 3.3.

3.2 Risk mitigation

Even though the distributed log provides traders with an overview of ongoing and finished trades, XChange does not yet address the situation where a trader conducts *counterparty fraud* during a trade. As a result, the economic gains of adversaries may be unbounded since a malicious trader can commit fraud in many trades. Consider a simple trade between Alice and Bob, where Alice sells 2 FabTokens for 40 XRP, and Bob sells 40 XRP for 2 FabTokens. Both Alice and Bob are expected to individually send their respective assets to each other. Since we do not assume atomic exchange, one of the parties, say Alice, has to initiate the first transfer. As soon as Alice sends 2 FabTokens, she is exposed to *counterparty risk*, as Bob *may not* send back the respective 40 XRP.

In this section we present risk mitigation strategies of XChange that limit the gains of traders committing counterparty. These strategies mainly aim at minimizing the assets at stake by dividing each trade into smaller chunks (Section 3.2.1) and by bounding the total amount of obligation a party can enter into (Section 3.2.2). Throughout the paper, we name the party that is exposed to counterparty risk as *risktaker*, while the other party is called *risky*. Determination of the trade roles (who becomes the risky party and who becomes the risktaker) in a prospective trade is explained in Section 5.

3.2.1 Incremental settlement

The first risk mitigation strategy we introduce is *incremental settlement*, where a trade is incrementally completed in k near-equal, smaller payments. Assume that in our fictitious trade between Alice (A) and Bob (B), parties agree to use an incremental settlement with $k = 2$. The total trade, therefore, would consist of four consecutive payments as illustrated in Figure 5. Alice is the *risktaker* in this trade and she does the first payment. Notice that

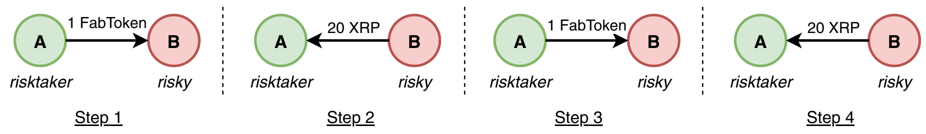


Figure 5 An asset exchange with $k = 2$ between Alice (A) and Bob (B), trading a total of 2 FabTokens and 40 XRP. During this trade, Alice is the risktaker since she is exposed to counterparty risk. Bob is the risky party since he is able to commit counterparty fraud after step 1 and 3

after each payment by Bob, the parties are on par with each other.[6] Termination of the trade at this state would not cause an economic loss for any of the parties. On the other hand, after each payment by Alice, the trade is in a state where Bob has an economic gain of 1 FabToken and Alice experiences an economic loss. With incremental settlement and $k = 2$, Alice is risking only a loss of 1 FabToken, instead of 2 FabTokens. We call the amount of risked assets as the *assets at stake*.

Similar to making multiple, smaller payments in the Interledger protocol, traders in XChange can gradually complete a trade in smaller steps and thus keep the risks manageable. On the one hand, in a trade where each party transfers value v to the counterparty, the economic gains of an adversary is reduced to $\frac{v}{k}$. On the other hand, incremental settlement prolongs the trade since more payments are made, and as such more transactions must be included on the blockchains that are managing the assets being traded. In general, a trade completed using incremental settlement requires $2k$ `Payment` records in the distributed log. In XChange the value of k is determined by the risktaker party of the trade and recorded in the `Agreement` record associated with the trade.

As we experimentally show in Section 8.2, incremental settlement reduces the value at stake during ongoing trades. This strategy is not applicable when a trade cannot be completed incrementally, e.g., when exchanging property titles or securities. Such assets are usually represented by fungible tokens on the ledger and gain their value from uniqueness. Even though these assets cannot be exchanged using incremental settlement, traders can still benefit from the second risk mitigation strategy that bounds the economic gains of adversaries.

3.2.2 Bounded obligations

Even though incremental settlement reduces the number of assets the risktaker puts at stake, it does not prevent an adversary from taking part in multiple concurrent trades as a risky party and commit counterparty fraud. In the simple trade example above, consider the case where Bob initiated another trade as a risky party with Charlie before finalizing his trade with Alice. Assume further that both trades are in a state where both Alice and Charlie have made their payments and are waiting for Bob's response. There is no restriction for Bob to enter into another trade before fulfilling its trade obligations to Alice and Charlie.

By devising rules that describe when a trader will start a trade with another party, we can bound the economic gains of adversarial parties under the assumption that non-adversarial traders follow the protocol. We notice that the *risky* party of a trade has no reason to

[6]We assume here that a trade exchanges an equal amount of value between both traders. In practice, there are usually small profit margins where one party would gain slightly more in value when the trade is complete.

refrain from engaging in an upcoming trade since it has nothing to lose. A trader becoming a risktaker in a prospective trade, however, must assess the risky party by inspecting the distributed log to determine if it is "safe" to engage in a trade with it.

One way to mitigate the risk of counterparty fraud would be to forbid a trader from being risky in simultaneous trades. However, this approach may lead to a situation where a risktaker can arbitrarily delay the trade duration, preventing the risky party to engage in trades with others. Instead, we choose to bound the *obligations* a trader enters into, by limiting the total amount of *assets at stake* within trades where a particular trader is involved in as a risky party. In other words, XChange employs trade restrictions which ensure that a malicious trader can only commit counterparty fraud up to a specific value.

In XChange, every trader a assigns a *trust threshold* $u_a(b)$ to every prospective trader b, and refuses to enter into the trade with b if the total amount of *assets at stake* in all *open* trades in which b is the risky party is larger than $u_a(b)$. Open trades are the ones which do not have a respective `Finalize` record for its `Agreement` record in the distributed log. Formally, given a distributed log $\mathcal{L}$, let $P_{\mathcal{L}}$ be the set of all open trades and $P_{\mathcal{L}}(b)$ be the set of open trades in which trader b is the risky party. Let $V(t)$ be the *assets at stake* of a trade t. This value represents how much value a risky party can seize during a trade. The total value of obligations of a trader b is referred to as $B(b)$ and is as follows:

$$B(b) = \sum_{t \in P_{\mathcal{L}}(b)} V(t) \tag{1}$$

A trader a accepts to be a risktaker in a prospective trade t' with trader b if the following holds:

$$u_a(b) \geq B(b) + V(t') \tag{2}$$

We illustrate the idea of bounded obligations in Figure 6 which shows three scenarios involving traders a, b and c. In all the scenarios, a trader a has to decide if it wants to start a prospective trade t_1 with trader b. We assume that the value of assets involved in trades can be expressed into another asset type, say in United States Dollars ($). This conversion could be based on the market price of the involved assets.[7] The value of *assets at stake* in prospective trade t_1 is $10, and both parties have agreed that a becomes the *risktaker* and b becomes the *risky* if the trade starts. Trader a determines a trust threshold $u_a(b) = \$15$ for trader b.

Assume b is already involved in another trade t_2 with the trader c and that t_2 is not finalized.

In Figure 6a, trader b has the role *risktaker* in t_2. Since b is not the risky party in t_2, it does not have any obligations, i.e., $B(b) = 0$. Therefore, as long as $u_a(b) > 0$, trader a can decide to start a trade with b. In Figure 6b and c, b is the risky party of t_2 where $V(t_2)$ is equal to $4 and $10, respectively. In Figure 6b, trader b's obligations stemming from ongoing trades amount to $4, i.e., $B(b) = 4$. Since b's prospective obligations $V(t_1) + B(b)$ is smaller than the trust threshold, a agrees to trade with b. In Figure 6c, the prospective obligations of b amount to $20 and thus exceed the threshold $u_a(b)$, which would result in the refusal of t_1 by trader a. However, even in this scenario, traders may agree to reduce *assets at stake* by increasing the k. Using $k = 2$, for example, lowers $V(t_1)$ to $5.

[7]It can also be that traders have differing opinions on the market price of a particular asset, e.g., based on their buy and sell orders.

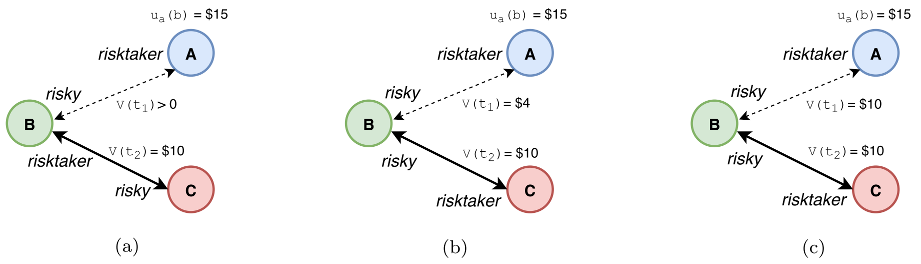


Figure 6 Three scenarios in which a trader a has to decide on starting a prospective trade t_1 in which b will become risky. a agrees with the trade in (**a**) and (**b**), and refuses to trade in (**c**). A solid line represents an ongoing trade whereas a dashed line represents a prospective trade

3.2.3 Further comments on risk mitigation

Flexible conformance We note that a trader can always ignore the risk mitigation strategies described in this section and engage in other trades *at its own risk*. Doing so, however, does not provide restrictions on the economic gains of adversarial parties but it enables participants to engage in trade with traders with which there is an existing trust relation (e.g., the traders know each other in real life). Trades that parties started at their own risk do not impact the obligations of the risky party in such trades. Such trades contain a special flag in the associated `Agreement` record.

Subjectivity and trust We note that the threshold function u_a is a subjective matter for a trader a and is highly dependent on the notions of trust and reputation, which are outside the scope of our work. Without loss of generality and for simplicity, we assume in the rest of the paper that $u_a(b)$ is equal to a constant U for all trader pairs a and b.

Determination of k Parameter k signifies the number of payments each party does in a trade. This value is proposed by the risktaking party during trade negotiations and is included in the `Agreement` record in the distributed log. We note that both sides of a trade is concerned with the value of k. For the risktaker, k determines the *assets at stake*, i.e., the value that the risktaker may lose in case of counterparty fraud by the other party. For the risky party, k affects the maximum rate of trade a party can be involved in as a risky party. While lowering the value of k brings together low-risk advantage for the risktaker and trust advantage for the risky party, it, in return, increases the duration of a trade, i.e., the number of transactions needed to settle the trade.

3.3 Cancellation of a trade

We note that a trade may never complete if a risktaker goes offline. The total amount of *assets at stake* in all such *stalled* trades in which a party b is a risky party may reach a point where no-one wants to trade with b, even if b is not at fault. Therefore, the ability of a trader b to trade with others may be forever restricted. To address this situation, we allow a risky party to explicitly cancel an ongoing trade by including a `CancelTrade` record in the distributed log. This record can only be included by the risky party, and is only acknowledged by other traders if (1) the risktaker is currently responsible for transferring assets to the risky party during the trade, and (2) at least some time Δ_t has elapsed since the last activity in trade t. The value of Δ_t should be well above the confirmation times of transactions submitted to the involved blockchains, to avoid the situation where one might consider a trade as stale while a transaction is still being finalized in the involved blockchain. When a trade is canceled, no other assets should be exchanged. After the risky partner canceled participation in a trade, it loses its risky status and can then participate in other trades.

We note that a risky party a can try to trick another party b into acknowledging a `CancelTrade` record by publishing a `Payment` record with a non-existent transaction identifier. Therefore, b needs to inspect the involved ledger to determine the validity of a `CancelTrade` record created by a. However, b might not have the appropriate credentials to read transactions on this ledger. Even though the `CancelTrade` transaction might be valid, we assume that b will not acknowledge the `CancelTrade` record when it cannot accurately determine its validity. We argue this is reasonable since this particular situation is likely to be infrequent. We also believe that this design decision does not significantly limit the efficiency of our mechanism.

4 System assumptions and threat model

We first discuss the XChange system model. This includes our assumptions on the blockchains that are managing the assets being exchanged, the requirements of the distributed log used by XChange, and the specifications of the underlying XChange network. We then present the threat model of XChange, and state the goals and capabilities of adversarial parties.

4.1 Blockchain, distributed log, and network specifications

The XChange mechanism coordinates asset exchange between permissioned blockchains. W.l.o.g., we denote the blockchains that are managing the assets being exchanged by $\mathcal{B}_a$ and $\mathcal{B}_b$ respectively. XChange only requires that $\mathcal{B}_a$ and $\mathcal{B}_b$ can represent assets and transfer assets to another owner. The consensus mechanisms deployed by $\mathcal{B}_a$ and $\mathcal{B}_b$ might be fundamentally different. We assume that for each involved blockchain, the fraction of adversarial parties is bound by the threshold necessary to ensure safety and liveness properties. In PBFT-based consensus algorithms, this threshold is usually $\frac{1}{3}$ of all nodes involved in the consensus algorithm [8].

XChange stores trade records in a distributed log, denoted by $\mathcal{L}$. We require that the entries stored by $\mathcal{L}$ are immutable and append-only. If entries in $\mathcal{L}$ would be mutable or can be removed, a trader could trick a counterparty into starting a trade, commit counterparty fraud, and remove all traces of the trade. Similar to how participation in $\mathcal{B}_a$ and $\mathcal{B}_b$ is explicitly approved, participation in $\mathcal{L}$ should be managed by an authority. We envision that a trader joining XChange re-uses the well-defined identity under which it participates in one of the permissioned blockchains. We remark that $\mathcal{L}$ can, for example, be realized through a blockchain with support for smart contracts.

Users in XChange participate in a peer-to-peer network, which is used to send point-to-point messages to other users. This network is particularly used during trade negotiation, as we further specify in Section 5. We assume that peers in the XChange network know the network addresses of other peers.

4.2 Peer model

We now elaborate on the assumptions of peers participating in XChange.

Each peer in the XChange network owns a cryptographical key pair consisting of a public and a private key. The public key of a specific peer is known to others and uniquely identifies it in the network. Their private key is used to digitally sign data such as records appended to $\mathcal{L}$, or outbound messages in the peer-to-peer network.

As we discussed in Section 4.1, the digital identity of each peer in the XChange network uniquely identifies a real-world user. Identity validation should be performed by a Registration Authority (RA), which is external to our system. The RA could be the same authority that approved participation in $\mathcal{B}_a$ or $\mathcal{B}_b$. We assume that the RA does not collude with traders in XChange. In XChange, well-established digital identities are necessary to prevent misbehaviors such as a Sybil Attack and a distributed denial-of-service attack [16, 49]. We also use verified identities for accountability purposes, where misbehavior in a trade can be traced back to a real-world persona.

Whereas existing work primarily focuses on *how* assets are exchanged, the XChange mechanism also includes primitives for traders to specify trade interest through orders, and to find trading partners that can fulfill these orders. We distinguish between *makers* and *takers*. A maker is a peer that creates a specific order, whereas a taker is a peer that fulfills an order. Makers introduce trading opportunities and liquidity to the XChange network. A peer in XChange can act as both maker and taker, for distinct orders. The maker-taker order model is also adopted by related protocols that enable the exchange of tokens on the Ethereum blockchain, namely 0x and AirSwap [42, 56]. System designers can also consider to leverage more advanced decentralized matchmaking solutions, e.g., as described in our prior work [11].

4.3 Threat model

Adversarial parties in XChange aim to *maximize their economic gains* by committing counterparty fraud in ongoing trades. Adversarial parties could attempt to append invalid records to $\mathcal{L}$, intentionally ignore incoming messages in the peer-to-peer network, or refuse to respond to messages during trade negotiation. They also could strategically ignore the risk mitigation strategies described in Section 3.2. We assume that adversaries cannot compromise the integrity of the distributed log $\mathcal{L}$ used by XChange and cannot undermine the security of the blockchains that are hosting the assets being traded, $\mathcal{B}_a$ or $\mathcal{B}_b$. We also assume that the cryptographic primitives used by all involved blockchains are secure (e.g., digital signatures cannot be forged) and that the computational capabilities of adversaries are bounded.

5 The XChange trading protocol

We now present the XChange trading protocol for asset exchange between permissioned blockchains and specify all operations conducted by peers that are participating in a trade. We assume the system and threat model described in the prior section. The protocol consists of four phases. In the first phase, makers specify their trade interest by appending new orders to the distributed log $\mathcal{L}$. During the second phase, takers negotiate with makers about orders they would like to fulfill and append an `Agreement` record to the distributed log when they reach an agreement. During the third phase, the maker and taker execute the trade by exchange assets through payments. The trade is finalized in the fourth phase with a `Finalize` record.

5.1 Phase I: Order creation and cancellation

During the first phase of the XChange protocol, makers create new orders and append these orders to $\mathcal{L}$, see Figure 7. When a trader intends to buy or sell some assets, it constructs a new order which we denote by O. O contains details on the quantity and the type of assets that the maker desires to buy and sell. The order creator provides this information as a two-tuple of asset quantities, also called an *asset pair*. The first asset quantity in the asset pair indicates the assets that the order creator wants, and the second asset quantity indicates what the order creator offers in return. An asset quantity is described by the combination of an integer value and a string that indicates the asset type. For example, if a trader intends to sell 2 FabTokens for 40 XRP tokens, it creates an order with asset pair (2 FabToken, 40 XRP).

O includes an integer value, k, that specifies the order creator's preference regarding the number of partitions each payment is divided in. As discussed in Section 3.2.1, one way how XChange reduces value at stake is by using incremental settlement. The inclusion of k in O indicates the risk that the maker is willing to take in an upcoming trade that fulfills O if the order creator would become the risktaker. Furthermore, O includes the address of the wallet in which the order creator wishes to receive assets from a prospective trader during an upcoming trade. By including this information, a taker knows to which address it should transfer its assets. This information is also used by other traders to verify if the maker has received assets from a taker.

After adding all required fields to O, the order creator serializes the order and embeds it in an `Order` record. The order creator then appends the `Order` record to $\mathcal{L}$. The order identifier can be determined by taking the hash of the record content, which we denote by $H(O)$.

A maker can cancel any of their non-expired orders that are not being fulfilled by an ongoing trade. This is achieved by the maker appending a `CancelOrder` record containing $H(O)$ to $\mathcal{L}$.

5.2 Phase II: Trade negotiation

During the second phase of the XChange trading protocol, a maker and taker negotiate a trade, see Figure 8. If the negotiating maker and taker agree to trade, one of the parties appends this agreement to $\mathcal{L}$. We now describe this negotiation process.

This phase starts when a taker discovers an order O, included on $\mathcal{L}$, that it wishes to fulfill. Assume that this order has been created by a maker M. Before sending a trade proposal

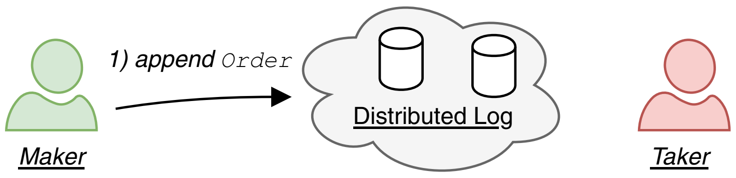


Figure 7 Phase I of the XChange trading protocol: makers (depicted in green) indicate trade interests by appending an `Order` record to the distributed log

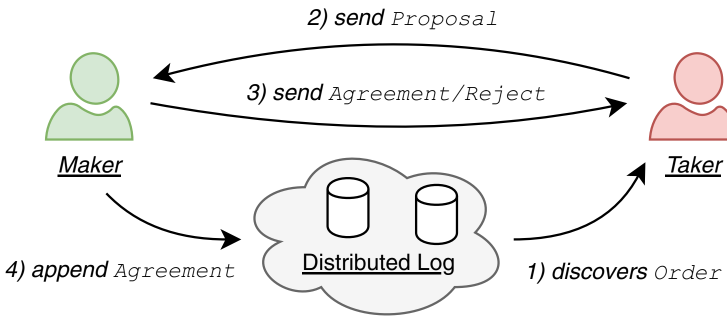


Figure 8 Phase II of the XChange protocol: a maker and taker negotiate a trade agreement. Upon a successful negotiation outcome, a dual-signed `Agreement` record will be appended to the distributed log

to M, the taker performs two checks that determine if the taker should trade with M. First, the taker checks if it is willing to trade with M as a person. For instance, M could have attempted to commit counterparty fraud in the past, which could be a reason for the taker to refrain from trading with M. Second, the taker determines if it is safe to trade with M, according to the *bounded obligations* strategy described in Section 3.2.2. The taker checks the trades in which M is involved by inspecting the latest records on $\mathcal{L}$ involving M. If M is already involved in a trade T, the information on $\mathcal{L}$ also reveals if M in T is a risky party or a risktaker.

When all three checks pass, the taker creates and sends a `Proposal` message to M. A `Proposal` message contains a proposal for M to fulfill order O. A taker includes four pieces of information in a `Proposal` message. First, it includes the identifier of O, so the maker knows which order the taker wants to fulfill (a trader could have created multiple orders). Second, the taker includes its destination wallet address to which M should send its assets during the trade. Third, the taker includes an integer value, k, that indicates how much risk the taker is willing to take if it would become the risktaking party. Finally, the taker includes a boolean value in the proposal indicating if the taker becomes a risktaker in the upcoming trade. At a high level, a `Propose` message represents a new order that indicates the taker's trade preferences.

When M receives a `Proposal` message from taker T, it also verifies whether it wants to trade with T. Specifically, M performs the same three checks as the taker did. Furthermore, M verifies if it agrees with the role classification proposed by the taker. If validation fails, the maker immediately sends a `Reject` message back to the taker, containing the identifier of the rejected order and, optionally, why M has rejected the proposal. If M agrees with the proposal and also wishes to trade with T, M constructs a `Agreement` record, which includes the identifier of the order being fulfilled and the proposal created by the taker (including the taker's signature). This `Agreement` record is signed by M, sent back to T, and appended to $\mathcal{L}$. Inclusion of the `Agreement` record on $\mathcal{L}$ binds the maker and the taker to the trade agreements. Since the risktaker is exposed to counterparty risk in the upcoming trade, the preferred value of k by the risktaker is used during the upcoming asset exchange. If the maker is the risktaker, the value of k as specified in the `Order` record describing O is leading. Otherwise, if the taker becomes the risktaker, the value of k specified by the taker in its proposal is leading.

5.3 Phase III: Trade settlement

During the third phase of the XChange trading protocol, assets are exchanged between the maker and taker, and the trade is settled. Figure 9 visualizes a trade between a maker and taker, with $k = 1$, where the maker sells FabTokens, a token managed by Hyperledger Fabric, and gets XRP (Ripple) tokens in return from the taker. This trade, fulfilling order O, consists of two payments, one from the maker to the taker, and one from the taker to the maker.

Asset exchange starts by the risktaker (the taker in this specific example) issuing a transaction to the Ripple network managing the XRP tokens. This Ripple transaction transfers XRP tokens from the wallet specified in the `Proposal` message to the wallet address that was specified by the maker in the `Order` record associated with O. After the taker has issued this transaction in the Ripple network, it appends a `Payment` record to $\mathcal{L}$, which contains the identifier of the order being fulfilled, and the identifier of Ripple transaction. The `Payment` record allows the maker (and other traders) to verify that the taker has transferred the correct amount of assets to the maker.

After the maker has verified that it received the agreed amount of XRP tokens, it conducts the next payment by issuing a transaction to the Hyperledger Fabric network. This transaction transfers FabTokens from the wallet specified in the `Order` record to the wallet that was specified by the taker in the `Proposal` message. The maker then appends a `Payment` to $\mathcal{L}$, which includes the identifier of the transaction in the Hyperledger Fabric network. This payment process repeats until all assets have been exchanged between the maker and the taker.

There is a risk that a trade does not progress when one of the traders becomes inactive. As pointed out in Section 3.2, a stale trade is only a minor concern for the risktaker since this party can still engage in other trades after Δ_T time has elapsed. A risky party can explicitly cancel an ongoing trade to dismiss its responsibility as a risky party by appending a `CancelTrade` record to $\mathcal{L}$. This record only contains the identifier of the order currently being fulfilled. Other traders should verify that the `CancelTrade` adheres to the rules as outlined in Section 3.2, to prevent the risky party from illegally canceling a trade after having committed counterparty fraud.

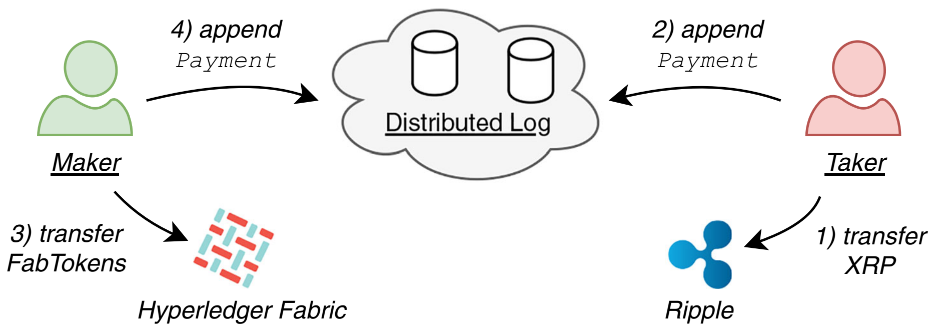


Figure 9 Phase III of the XChange protocol: a maker and taker trade by exchanging assets. In this trade, the taker is the risktaker (initiating the first payment) and the maker is the risky party

5.4 Phase VI: Trade finalization

When all assets have been exchanged, the party receiving the final payment during a trade creates a `Finalize` record and appends it to $\mathcal{L}$, see Figure 10. Since the risky party conducts the final payment during a trade, finalization is always performed by the risktaker. Inclusion of a `Finalize` record on $\mathcal{L}$ completes a trade, say T_1, between the maker and taker, and both parties can now start new trades with others.

6 Security analysis

We now analyze the security of the XChange mechanism. First, we prove that the economic gains of adversarial parties committing counterparty fraud are upper-bounded. We then discuss the scenario where multiple adversaries collude to gain an advantage as a group.

6.1 Counterparty fraud limitations

We further analyze the *bounded obligations* strategy presented in Section 3.2.2. This strategy define an upper bound on the obligations an adversary can enter into, under the condition that all honest traders act rationally and try to minimize their risk.

Limiting the gains of adversaries To show that XChange limits the amount of fraud, we assume —for clarity— that all honest peers fix a unit trust threshold U, and the number of payments in each trade t is fixed to K. Specifically, where P is the set of traders and T is the set of all trades, the following is assumed:

$$(u_i = U) \quad \forall i \in P \qquad \text{and} \qquad (k_j = K) \quad \forall j \in T.$$

Under these two assumptions, XChange guarantees that:

(1) the total value of assets an adversary can gain as result of counterparty frauds is limited to U,
(2) the loss of an honest party in a trade is limited to V/K, where V is the *assets at stake* during the trade.

Assume an adversarial trader B is involved in $(n-1)$ trades denoted by $t_1, t_2, \ldots, t_{n-1}$ in which B is the risky party and is in trade negotiations with an honest trader A for the prospective trade t_n. Assume $\sum_{i\in\{1,\ldots,n-1\}} V(t_i) \leq K \cdot U$ and $\sum_{i\in\{1,\ldots,n\}} V(t_i) > K \cdot U$. Under these conditions, trader A does start a trade with B, given that A follows XChange protocol.

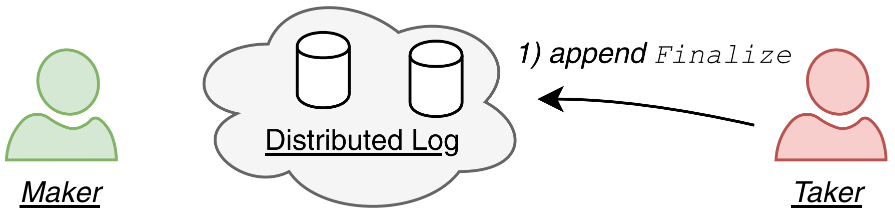


Figure 10 Phase IV of the XChange protocol: the taker finalizes the trade

We can now show the correctness of the statement (1) and (2) above. There are two ways a malicious party can commit counterparty fraud. Firstly, it can choose to become inactive in a trade and not conduct a payment back to the risktaker. Secondly, B could append a `Payment` record to $\mathcal{L}$ that points to a non-existent or invalid transaction, attempting to trick the risktaker counterparty and other traders. In both cases, the value of fraud in a trade t cannot exceed $V(t)/K$, which verifies statement (2). Therefore, even when we assume that B commits counterparty fraud in all the active trades it is involved, the total value of assets B can gain does not exceed U, which verifies statement (1).

We can now relax our assumptions on the objectivity of trust threshold (U) and the number of payments (K). Assuming each trade t has its own number of payments k_t agreed by the trading parties, the amount of assets the risktaker can lose in t is limited by $V(t)/k_t$. When $u_A(B)$ is the trader A's subjective trust threshold assigned to trader B, then trader A does not start a trade t with B if the existing obligations of B exceed $u_a(b) - V(t)/k_t$. Assuming k_t is not bounded and that $V(t)/k_t$ may converge to zero, a trader B can start a trade as a risky party with a trader b, only if its obligations occurred from ongoing trades is limited by $u_a(b)$. Accordingly, defining $\bar{u} = \max\{u_j(b) : j \in V\}$ where V is the set of all prospective traders with b, the total amount of assets that b can seize in XChange cannot exceed $\bar{u}$.

Limit on the risktaker's loss In XChange, the risktaker's loss in a single trade is bounded with the trust threshold u associated with the risky party. Furthermore, as the risktaker is involved in the determination of number of payments (k) during trade negotiations, it can reduce its own risk to any extent. Nevertheless we note that XChange does not introduce a theoretical bound on the loss of a risktaker over time, but delegates the risk management to the risktaker by assuming a trust mechanism to determine the trust threshold. Specifically, XChange provides the risktaker with two important parameters u and k with which the risktaker can minimize its own risk, relying on the trust mechanism.

Malevolent cancellation of a trade We now analyze the situation where a risky party B cancels its ongoing trade t_1 with A by appending a `CancelTrade` record to $\mathcal{L}$, before the trade is finalized and when it is B's turn to make payment. When a third party C considers entering in trade t_2 with B, it will discover the `CancelTrade` record in $\mathcal{L}$ and check whether the trade cancellation by B is legitimate (see Section 3.3). Recall that the cancellation of a trade by B is legitimate if it is currently the responsibility of the risktaker to conduct the next payment. The trade cancellation of t_1 is therefore not valid since B has committed counterparty fraud. If the trade cancellation were legit, B would not have been under no obligation in trade t_1, since it would have transferred assets to A. Now, when C verifies the status of B and detect an illegitimate trade cancellation, C will not engage in trade t_2 with B.

6.2 Collusion resistance

In a collusion attack, a group of traders follows a common strategy to subvert the network or gain advantages as a collective. The XChange mechanism is highly resistant against collusion attacks since adversarial parties are not able to gain more economic gains when working together, given that $\mathcal{L}$ provides secure storage of included trade records. We argue that the resistance against collusion can be addressed to the absence of group-based coordination in XChange. Tasks that would involve coordination among a group are usually vulnerable to attacks where a majority of the group colludes to gain advantages over non-colluding users.

In XChange, trade proceeds through the direct interaction between the involved traders and therefore, cannot be influenced by groups of colluding adversaries.

7 Distributed logging of trade records

The XChange trading protocol described in Section 5 requires a distributed log to securely and irreversibly store `Order`, `CancelOrder`, `Agreement`, `Payment`, `Finalize` and `CancelTrade` records. We choose to build XChange upon TrustChain [41] which is a shared data structure with a sharp focus on tamper-resilience and trustworthy record storage. In this section, we motivate our choice for TrustChain and outline how TrustChain is used to store XChange records.

7.1 TrustChain: A scalable ledger for accounting

Based on the idea of blockchain ledgers that order transactions in a directed acyclic graph (DAG), Otte et al. designed, implemented, and deployed TrustChain. TrustChain is a ledger that is optimized for lightweight, tamper-proof accounting of data elements [41]. The key idea is that individuals maintain and grow their *individual ledger* with records. Other users verify these records according to some pre-defined rules. This makes TrustChain similar to solutions for tamper-proof, distributed logging, such as PeerReview [22]. TrustChain does not aim to prevent integrity attacks on the data structure, e.g., fork creation, but instead guarantees eventual *detection* of these attacks. This yields superior scalability compared to other ledgers but allows for the situation where some malicious activity targeted at the ledger might go undetected for some time, for example, the hiding of specific transactions. In TrustChain, this can be addressed by waiting longer before accepting a record as valid. Individuals in TrustChain are not required to store all records in the network and might choose to store different parts of the global DAG ledger. TrustChain does not reach a global consensus over all records but relies on participants to detect inconsistencies in individual ledgers.

We argue that TrustChain is a suitable ledger to store XChange records, for the following four reasons. First, TrustChain allows participants to verify the integrity of other individual ledgers themselves, and determine whether a party is already involved in a trade or not. There is no requirement to reach a global consensus on the integrity of included records. Second, TrustChain does not require network-wide replication of all records but enables individuals to selectively share parts of their individual ledger with others. This feature reduces storage requirements and allows XChange to also run on devices with storage limitations, as we demonstrate in Section 8.3. Third, the TrustChain structure is optimized to store bilateral records that are signed by two parties. This aligns well with the XChange trading protocol since many operations could benefit from support for bilateral records (for example, trade agreements). Finally, TrustChain is already being used by various decentralized applications that require accounting features, such as self-sovereign identities and inter-bank payments [12, 50]. At the time of writing, the public TrustChain ledger contains over 160 million records, created by 96’000 unique identities.[8]

[8] See http://explorer.tribler.org

7.2 Storing TrustChain records

We now outline how a record between two interacting users A and B is recorded in TrustChain, see Figure 11. Each record is stored within a block. Figure 11a highlights one block containing a record between A and B. Each block contains a single record (R). A record can be a generic description of any interaction between users, for instance, a trade agreement or a payment. Both interacting parties digitally sign the block with the record by using any secure digital signing algorithm. These signatures are included in the block and ensure that participation by both parties is irrefutable. It also confirms that both parties agree with the record itself. Others can effectively verify the digital signatures included in a block. After all required signatures have been added to a block, the block is committed to the local databases of the two interacting parties and broadcast to a limited number of random peers in the network.

The security of stored blocks is improved by linking them together, incrementally ordered by creation time. In particular, each block is extended with a description (hash) of the previous block. Each block has a sequence number that indicates its position in the individual ledger. This results in the structure shown in Figure 11b. As a result, each user maintains their individual ledger, which contains all records in which they have participated. This sets TrustChain apart from the structure of traditional blockchains, where the entire network maintains a single, linear ledger.

Note how the blockchain structure in Figure 11b allows A to modify blocks in their individual ledger without being detected by others. In particular, A can reorder the blocks in its individual ledger since validity can quickly be restored by recomputing all hashes. In most blockchain applications, the global consensus mechanism prevents this kind of manipulation. TrustChain uses a more efficient approach: each block is extended with an additional

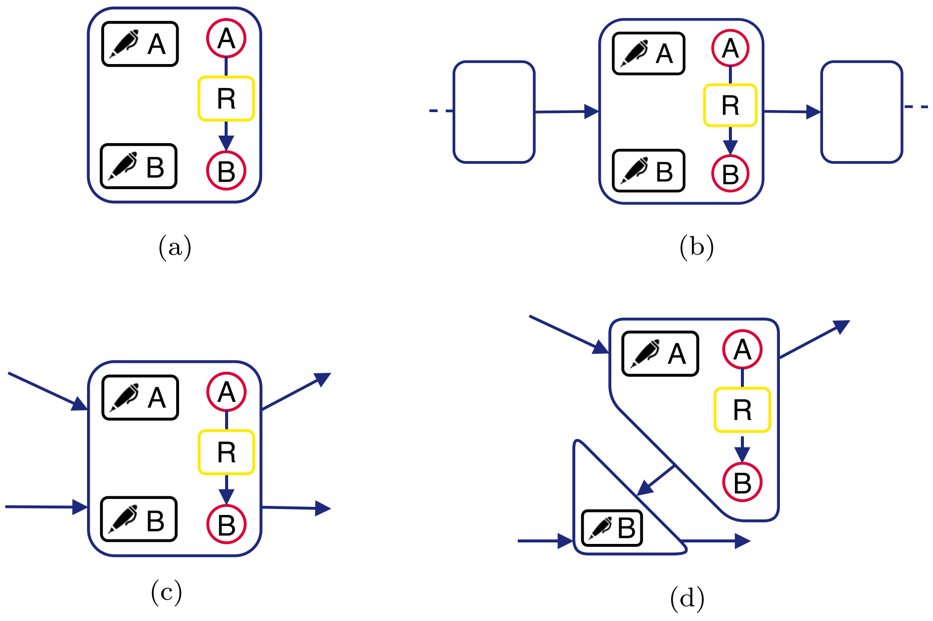


Figure 11 Storing records in TrustChain

(hash) pointer that points to the previous block in the individual ledger of the counterparty. This is visualized in Figure 11c. Each block now has exactly two incoming and two outgoing (hash) pointers, except for the last block in an individual ledger, which only has two incoming pointers. Modifications of the individual ledger by A, like reordering or removing blocks, can now be detected by one or more counterparties. To prove this fraud, a counterparty reveals both the correct block and the invalid block created by A.

When two parties transact and create a block, their chains essentially become entangled. When users create more records with others, it leads to the directed acyclic graph (DAG) structure, as shown in Figure 12. Figure 12 shows seven blocks, created by seven unique users. Each block is added once to the individual ledger of all parties involved in the record. For a more advanced analysis of the technical specifications and security of TrustChain, we refer the reader to the original paper by Otte et al. [41].

7.3 Improving TrustChain scalability

According to Otte et al., TrustChain is designed to scale [41]. However, we identify that its design limits a user to one pending block creation at once. The main issue is that the digital signature of a counterparty is required before a new block can be appended to an individual ledger (since the input for the hash of each new block includes all signatures in the previous block). This enables an attack where a malicious user can purposefully slow down the block creation of others by delaying the signing process of a bilateral transaction it is involved in. It also limits the growth rate of individual ledgers and reduces the overall scalability of TrustChain.

We contribute to TrustChain and improve its scalability by adding support for *concurrent block creation*. The idea is to remove the requirement for a digital signature of the counterparty when appending new blocks to an individual ledger. We believe that this concurrency is necessary since it allows traders to append new records without reliance on other parties.

Our solution is visualized in Figure 11d. It shows a record between users A and B, initiated by A. We partition a block in two parts, and each block partition is appended to the individual ledger of exactly one party. Construction of a block between A and B now proceeds as follows: first, user A creates a record by constructing a block partition with the

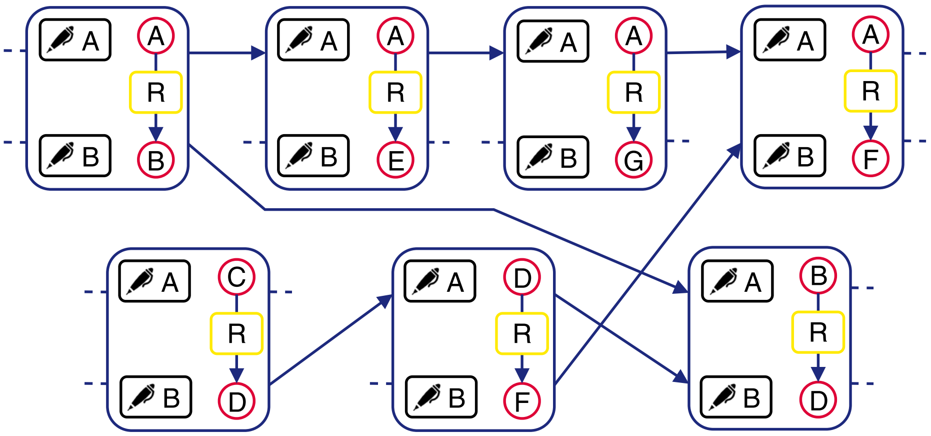


Figure 12 The TrustChain ledger, with seven blocks created by seven participants

record content and its digital signature. User A adds this block partition to its individual ledger immediately (note that it does not include the digital signature of B). A now sends the block partition to B. If B agrees with the transaction, it signs the block partition created by A, adds it to its individual ledger, and sends his block partition (with their signature) back to A. User A stores the block partition created by B in its local database. The participation of both parties in this record can now be proven with both block partitions. This mechanism allows users to be involved in multiple block constructions at once.

7.4 Logging trade records on TrustChain

We now outline how `Order`, `CancelOrder`, `Agreement`, `Payment`, `Finalize` and `CancelTrade` records are stored on TrustChain. Figure 13 shows a part of the TrustChain ledgers of traders A and B. It includes a sequence of records that indicate a finished trade between A and B. Trade agreements, created during the second phase in the XChange protocol, are stored within a bilateral `Agreement` record and digitally signed by both involved traders. Individual payments are stored within bilateral `Payment` records. A `Payment` record signed by both parties indicates that the payer has conducted the payment and that the payee has observed the payment. Finally, a trade finalization is stored within a bilateral `Finalize` record. Since the overhead of creating new TrustChain records is low, we also store orders as unilateral `Order` records in individual ledgers. A unilateral record only contains the digital signature of its creator. Figure 13 shows a `Order` record, created by maker A. Furthermore, `CancelOrder` and `CancelTrade` records are also included as unilateral records in one's individual TrustChain ledger.

A particular issue is that the fragmented nature of the TrustChain DAG makes it difficult for takers to discover interesting orders quickly. Specifically, `Order` records are by default only stored in the individual ledger of the order creator. Therefore, we introduce *matchmakers*, peers that continuously collect the TrustChain records of other peers in the network, and organize the information in `Order` records in a local database. Matchmakers aggregate orders, and takers can query the database of matchmakers to find interesting orders. Makers also send their TrustChain blocks with an `Order` record to known matchmakers after creation. The role of matchmakers in XChange is comparable with that of relay nodes in 0x [56] and indexers in AirSwap [42].

8 Implementation and evaluation

In this section, we present the implementation of XChange and our experimental evaluation. The evaluation answers the following three questions: (1) how effective is XChange at reducing fraud gains? (2) what is the overhead of XChange, in terms of trade duration,

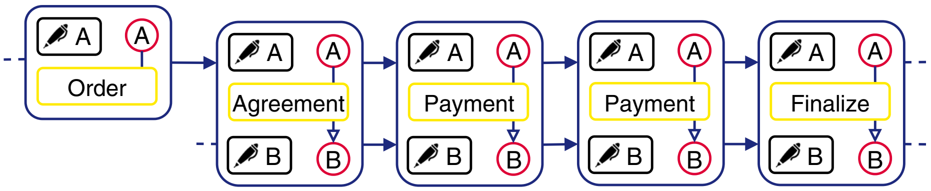


Figure 13 A part of the TrustChain ledger, storing an order created by a maker A, and full specifications of a finished trade between A and B

when XChange is deployed on low-resource devices? And (3) How scalable is XChange in terms of throughput and trade duration when increasing the system load?

The following experiments quantify the effectiveness, performance and overhead of our XChange mechanism. During these experiments, we assume that asset settlement is instant. As such, we do not actually connect a permissioned ledger to XChange for asset transfers. We believe this is a reasonable experiment setup since our aim is to evaluate the scalability and overhead of our approach without the interference of external systems.

8.1 Implementation details

We have implemented the XChange in the Python 3 programming language. Our implementation spans a total of 4’702 lines of code and uses an event-based programming model, powered by the built-in `asyncio` library. The implementation is open source and all software artifacts (source code, tests, and documentation) are published on GitHub.[9]

Networking We have built XChange on top of an existing networking library that is also used by TrustChain. This library provides the functionality to devise decentralized overlay networks and has built-in support for authenticated network communication, custom message definitions, and UDP hole punching.[10] For efficiency reasons, the UDP protocol is used for message exchange between peers.

Request stores To correctly process incoming messages during trade negotiation (phase II of the XChange protocol, see Section 5), XChange stores the state of outgoing messages. The state of outgoing messages is stored in distinct *request stores*. For each outgoing message that has a state attached, a unique identifier is generated, a new request store containing this identifier is created and the generated identifier is appended to the outgoing message. Traders that receive a message with this identifier are required to include the same identifier in their response message. Incoming response messages with an unknown identifier are discarded and not processed further. Each request store can have an optional timeout, indicating the duration after which the request store times out. When a request store times out, it is deleted.

Wallets XChange organizes different types of assets within wallets. These wallets provide a convenient interface to the information provided by connected blockchain platforms. Wallets expose functionality to query the existence of specific transactions, fetch the content of specific transactions, and to transfer available assets to another trader.

Our implementation contains a `Wallet` base class that can be extended by programmers to create wallets that store different types of assets. For testing purposes, we have implemented a `DummyWallet`, which is used when executing the unit tests and when running the experiments described in this section. This wallet does not interact with any blockchain and simply waits for some duration before returning a (fake) response.

[9] See https://github.com/tribler/anydex-core
[10] See https://github.com/tribler/py-ipv8

8.2 Reducing fraud gains

Our first experiment quantifies the effectiveness of reducing fraud gains when trading with XChange. We experimentally show the effectiveness of the risk mitigation strategies discussed in Section 3.2.

Setup and workload For this experiment, we reconstruct a real-world dataset, containing buy and sell orders published on the BitShares blockchain [46]. The BitShares platform enables users to issue custom assets and to trade these assets with others. We extract the buy and sell orders made during the last week at the moment of writing, and replay them with XChange. This results in a dataset with 230’000 orders, consisting of 125’527 buy orders, 104’423 sell orders and 212’489 cancellation events of existing orders. These orders have been created by 1’161 unique users and involve 243 different assets. Our data includes the orders created between November 11, 2020, and November 18, 2020. Since our dataset does not contain granular temporal information on order creation and cancellation, we assume that each order is uniformly created in the time interval between the last block and the block that contains this specific event. To accurately apply the *bounded obligations* strategy (see Section 3.2.2), we compile a list with the market price of all assets in our dataset, expressed in USD, by crawling a major BitShares block explorer.[11] We were unable to accurately determine the market price for 36 assets since they have a low or zero trading volume. We ignore the orders trading such assets during our experiment. We have published all scripts to construct this dataset in a separate GitHub repository.[12]

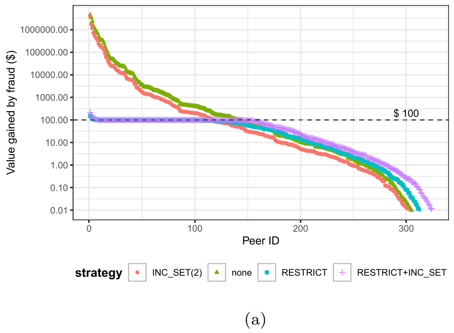
Since it is impractical to replay all the events in our dataset in real-time, we substitute our networking library with a custom discrete event simulator that is fully compatible with the `asyncio` library. At the start of the experiment, we create wallets for all peers with an unlimited amount of assets. To test the limitations of our mechanism in a highly adversarial setting, we model all peers as fraudsters, where they steal incoming assets whenever possible. Specifically, they commit counterparty fraud by not issuing a subsequent asset transfer after receiving some assets. During our experiment, a single peer acts as matchmaker and notifies traders about opportunities for their buy and sell orders. We fix the trust threshold U to \$100 for all peers when the *bounded obligations* strategy is enabled, meaning that the economic gains of an adversary are at most \$100.

We test the effectiveness of our mechanism with combinations of the two risk mitigation strategies discussed in Section 3.2. With the `INC_SET(k)` strategy, we refer to the incremental settlement strategy where each trader makes k payments to the counterparty during a single trade. The `RESTRICT` strategy denotes the strategy where a trader follows the *bounded obligations* strategy to verify whether it should trade with another party or not (see Section 3.2). We consider four experiment settings in total, with combinations of the `RESTRICT` and `INC_SET` strategies, and when no risk mitigation strategy is active. We note that the number of incremental payments when enabling the `RESTRICT` strategy is not fixed and depends on the current and prospective obligations of a counterparty.

Results We show the results of our fraud experiments in Figure 14. Figure 14a shows the economic gains of adversaries, for combinations of the risk mitigation strategies discussed in Section 3.2. We show the value gained by adversaries on a logarithmic vertical axis.

[11] See https://cryptofresh.com/api/docs

[12] See https://github.com/devos50/bitshares-orderbook-scripts



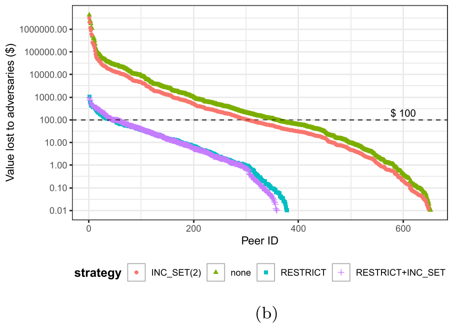


Figure 14 The economic gains of adversaries and the losses of traders when replaying 230’000 BitShares orders with XChange, for different risk mitigation strategies. We have fixed the trust threshold U to \$100

During our experiment we keep track of the fraud committed by adversaries, and sort these peers by the amount of fraud they have committed in USD. The horizontal axis shows the identifier of these peers. The total fraud gain without any risk mitigation strategy is \$18.5 million. This number is reduced to just \$18’609 under the `RESTRICT+INC_SET` strategy, *a reduction of 99.9%*. Under the `RESTRICT` strategy, the total fraud gain is \$16’260, lower than the gains under the `RESTRICT+INC_SET` strategy. We address this due to the fact that some trade proposals are being denied since they cannot be completed without incremental settlement; these trades, however, might have been possible when using incremental settlement, which would have resulted in more fraud instances. We also note that a few adversaries have committed fraud with a total value of over \$1 million when running without any risk mitigation strategy, and under the `INC_SET(2)` strategy. These successful adversaries likely created orders with competitive market prices, resulting in more trade proposals and opportunity for fraud.

Figure 14a clearly shows the effectiveness of the *bounded obligations* strategy. However, we observe that a few adversaries were able to commit fraud with a total value that exceeds our bound of \$100. For the `RESTRICT+INC_SET` strategy, twelve adversaries have committed fraud with a total value over \$100. We have identified that this issue arises from the weak consistency guarantees by TrustChain. Specifically, a trader A can be involved in the negotiation about many other orders at the same time, which together would exceed the bound of \$100. When all counterparties query the individual ledger of A around the same time, all these parties might decide that it is safe to trade with A and as a result engage in trade with A. In addition, a counterparty might deliberately refrain from send its latest record(s) back, which we refer to as the *record withholding attack*. To address these issues, we suggest two extensions to XChange and TrustChain. First, XChange can also record all the trade proposals, and their responses by counterparties (accept or reject). Counterparties can take the outstanding trade proposals into consideration when applying risk mitigation. To address the record withholding attack, a party can disseminate the latest record of its counterparty in Distributed Hash Table (DHT), e.g., Kademlia [34]. Traders can then query the DHT network to fetch the latest record of a party A, and then query the individual ledger of A up to that record. Even with this timing issue present, Figure 14a still shows that our risk mitigation is highly effective and is capable of significantly reducing fraud gains.

Figure 14b shows the economic losses of traders, for different risk mitigation strategies. Again, we notice that the *bounded obligations* strategy significantly reduces the economic losses of traders. The maximum individual loss when applying no risk mitigation strategy and under the `RESTRICT+INC_SET` strategy is \$4’255’731 and \$928.79, respectively. Figure 14b also highlights the effectiveness of incremental settlement under the `INC_SET(2)` strategy, compared to when no risk mitigation strategy is applied. In comparison to the fraud gains by adversaries, the economic losses by individual traders are not bounded but they are manageable.

Conclusion Our experiment with real-world data proves that the risk mitigation strategies by XChange are effective and significantly reduce fraud gains of adversaries. In particular, we have experimentally proven that incremental settlement indeed decreases fraud losses, and that bounded obligations bounds the economic gains by adversaries.

8.3 Trading on low-resource devices

Our second experiment quantifies the latency added by XChange when conducting a trade between two low-resource devices.

Setup and Workload This experiment is conducted with two hosted Raspberry Pis (3rd generation, model B+). The devices run the Raspbian Stretch operating system and the Python 3.5 interpreter. One device assumes the identity of trader A, and the other device acts as trader B. Furthermore, one device creates a new order, and the other device fulfills the order. The experiment is executed in an isolated environment: there is only network communication between the two Raspberry Pis. For this experiment, we use two different subclasses of `DummyWallet`, representing different assets. To measure the overhead of XChange, we configure these wallets such that assets instantly arrive when being transferred to another wallet. During the experiment, we log the timestamp of several events. At $t = 0$, the maker creates a new order. The trade is finished when both trading parties have signed a `Finalize` record and have committed this record to their individual ledgers.

Results Figure 15 shows a timeline of the events during a single trade between the two Raspberry Pis. The full trade sequence, from the moment of order creation to mutual possession of a dual-signed `Finalize` transaction, completes in 493 milliseconds, less than half a second. Almost half of the trade duration, 254 milliseconds, is spent in phase II of the XChange trading protocol, the trade negotiation phase. During this phase, a trader determines whether a counterparty is already involved in a trade by inspection of the records in the TrustChain ledger of the other party.

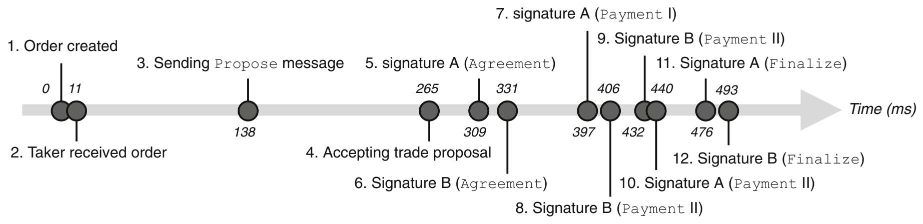

1. Order created
2. Taker received order
3. Sending Propose message
4. Accepting trade proposal
5. signature A (Agreement)
6. Signature B (Agreement)
7. signature A (Payment I)
8. Signature B (Payment II)
9. Signature B (Payment II)
10. Signature A (Payment II)
11. Signature A (Finalize)
12. Signature B (Finalize)
0
11
138
265
309
331
397
406
432
440
476
493
Time (ms)

Figure 15 A timeline of the events during a single trade between a maker A and a taker B. The experiment is conducted on two hosted Raspberry Pis (3rd generation, model B+). The total duration of the trade is 493 milliseconds

Conclusion This experiment shows that a full trade, including order creation, can be completed within half a second on low-resource devices if asset transfer would be instant. Based on this experiment, we argue that the deployment of XChange in an Internet-of-Things (IoT) environment would be viable since its communication and transaction creation overhead is minimal. Asset management is a common feature in IoT [21]. XChange can be used to coordinate asset exchange between different IoT environments. However, our trading protocol requires periodic inspection of blockchain during an ongoing trade. Since maintaining a full transaction history is not realistic given the storage restrictions of IoT devices, XChange should rely on dedicated full nodes that have to appropriate credentials to participate in a specific blockchain. We believe that devices with less processing capabilities than Raspberry Pis are still capable of maintaining and securing TrustChain records. This belief should be verified with further experimentation through a small-scale deployment of XChange in an IoT environment where blockchain-based assets are managed and traded.

Even though the low trade duration on low-resource devices is a promising result, the experiment is not representative of a realistic trading environment where there are many traders creating orders and exchanging assets simultaneously. Furthermore, the prior experiment does not reveal the impact of our risk mitigation strategies on performance. Therefore, our next experiment focuses on the scalability of XChange and shows how our mechanism behaves under a higher system load.

8.4 Scalability of XChange

We now perform scalability experiments to quantify the performance of XChange as the system load and network size increases.

Setup and workload To explore the limitations and overhead of XChange, we conduct scalability experiments on our university cluster. The detailed specifications of the hardware and runtime environment can be found online.[13] Our infrastructure allows us to reserve computing nodes and deploy instances of XChange on each node. We use the Gumby experiment framework to orchestrate the deployment of XChange instances onto computing nodes and to extract results from experiment artifacts.[14] The scalability experiment is controlled by a *scenario file*, a chronologically ordered list of actions which are executed by all or by a subset of running instances, at specific points in time after the experiment starts. Each run is performed at least five times, and the results are averaged.

We increase the system load, namely the number of new orders being created every second. As the system load grows, so does the number of traders in the network. We devise a synthetic dataset to determine the performance of XChange under a predictable arrival rate of orders. In a network with n peers running XChange, n orders are created every half a second. To avoid the situation where all instances create new orders at the same time, the starting time of this periodic order creation is uniformly distributed over all peers, based on their assigned IDs (ranging from 1 to n). Each peer acts as a matchmaker and sends a new order to four matchmakers, which each peer randomly selects when the experiment starts. The experiment lasts for 30 seconds, after which $30n$ orders are created in total. Each order buys a single token in return for another token, to make matchmaking a predictable process. After 30 seconds, the experiment is terminated.

[13] https://www.cs.vu.nl/das5/

[14] https://github.com/tribler/gumby

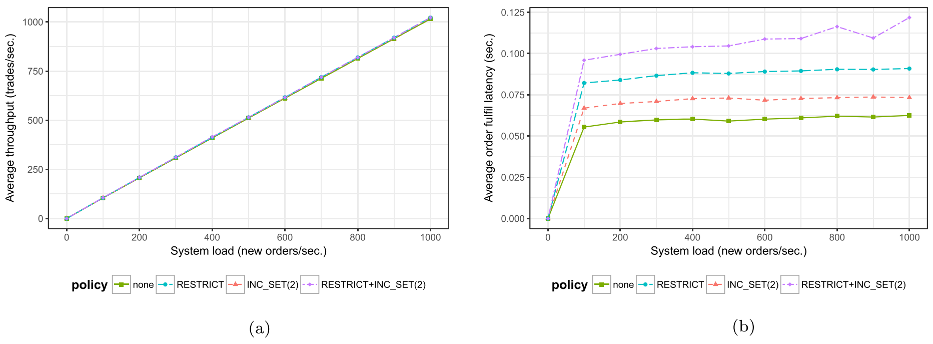


Figure 16 The peak throughput and order fulfill latencies as the system load increases

Scalability is measured as follows: first, we analyze the peak *throughput* observed during the experiment, in terms of trades per second. Second, we consider the average order fulfill *latency*, which is the time between the creation of an order and the time until this order has been completed (the order creator has exchanged all assets as specified in the order).

Results The results of the scalability experiments are presented in Figure 16. We run each experiment with a specific system load up to 1.000 deployed instances (which is close to the limitations of the used hardware). Figure 16a shows how the peak throughput (expressed in trades per second, vertical axis) behaves with respect to the system load (horizontal axis). All experiment settings hint at linear scalability as the system load increases. Furthermore, enabling risk mitigation strategies does not appear to have a notable effect on the peak throughput. Experimentation on more compute nodes should reveal whether this trend continues when the system load exceeds 1.000 new orders per second.

Figure 16b shows the average order fulfill latency when the system load increases, for the four risk mitigation strategies. The average order fulfill latency remains largely constant when the system load grows. Applying the restriction and incremental settlement strategies increases the average order fulfill latency, since more operations have to be performed to successfully complete an order. We observe a moderate increase of latency when applying the `RESTRICT+INC_SET` strategies when the system load grows to 1.000 trades per second. The high system load is likely to increase the duration of individual trades beyond 0.5 seconds, which means that the `RESTRICT` strategy prevents traders from initiating a new trade with others. Since a trader now has to find a new party to trade with, the average order fulfill latency increases.

Conclusion The main finding of this experiment is that the throughput (trades per second) scales linearly with respect to the system load and network size. We also observe that the average order fulfill latency remains largely constant as the system load grows. Further experimentation should reveal whether these trends continue with an even higher system load.

9 Conclusions

We have presented XChange, a universal mechanism for asset exchange between permissioned blockchain. XChange facilitates asset exchange without relying on particular transaction types or trusted third parties to mediate in the trading process. XChange records the initiation of a trade, individual payments, and the completion of a trade in a distributed log. By devising a set of rules that define when a party should engage in a new trade, we have limited the economic gains of adversarial parties. Specifically, when an adversary commits counterparty fraud, any further trade with this adversary are refused by honest parties until the fraud is resolved. Incremental settlement further reduces economic gains by splitting each payment into multiple, smaller ones.

We have implemented XChange and open-sourced its implementation. By replaying a dataset containing orders published on the BitShares blockchain, we have showed that XChange can significantly reduce fraud gains. We have also demonstrated the viability of trading on devices with low hardware capabilities. A single trade can be completed within half a second if asset transfers on external blockchain platforms would finish instantly. With a scalability experiment on our compute cluster, we achieved over 1’000 trades per second and found that the throughput of XChange in terms of trades per second scales linearly with the system load and network size.

We end by highlighting two promising research directions for further work. First, it would be helpful to extend our mechanism with privacy-enhancing features that do not reveal full trade details to the network. Second, our mechanism would benefit from a more extensive risk model that allows a trader to further reason about other traders while judging their trade proposals. This risk model can, for example, also take into consideration the “response time” of a counterparty and classify slower counterparties as riskier.

Funding This work was funded by NWO/TKI grant 439.16.614.

Code Availability (Software Application or Custom Code) The source code of XChange can be found at https://github.com/tribler/anydex-core.

Declarations

Conflict of Interests All authors are associated with Delft University of Technology.

References

1. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al.: Hyperledger fabric: A distributed operating system for permissioned blockchains. In: Proceedings of the Thirteenth EuroSys Conference, pp. 1–15 (2018)
2. Asokan, N., Schunter, M., Waidner, M.: Optimistic protocols for fair exchange. In: Proceedings of the 4th ACM Conference on Computer and Communications Security, pp. 7–17 (1997)
3. Back, A., Corallo, M., Dashjr, L., Friedenbach, M., Maxwell, G., Miller, A., Poelstra, A., Timón, J., Wuille, P.: Enabling blockchain innovations with pegged sidechains. http://www.opensciencereview.com/papers/123/enablingblockchain-innovations-with-pegged-sidechains **72** (2014)

4. Bentov, I., Ji, Y., Zhang, F., Breidenbach, L., Daian, P., Juels, A.: Tesseract: Real-time cryptocurrency exchange using trusted hardware. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 1521–1538 (2019)
5. Borkowski, M., McDonald, D., Ritzer, C., Schulte, S.: Towards Atomic Cross-Chain Token Transfers: State of the Art and Open Questions within Tast. Distributed Systems Group TU Wien (Technische Universit at Wien) Report (2018)
6. Brown, R.G.: Introducing r3 corda™: A distributed ledger designed for financial services. R3 Blog **5** (2016)
7. Buterin, V.: Chain interoperability. R3 Research Paper (2016)
8. Castro, M., Liskov, B., et al.: Practical Byzantine Fault Tolerance. In: OSDI, vol. 99, pp. 173–186 (1999)
9. Culwick, A., Metcalf, D.: The blocknet design specification (2019)
10. De Angelis, S., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A., Sassone, V.: Pbft vs proof-of-authority: Applying the cap theorem to permissioned blockchain (2018)
11. de Vos, M., Ishmaev, G., Pouwelse, J.: Match: A decentralized middleware for fair matchmaking in peer-to-peer markets. In: Proceedings of the 21th International Middleware Conference (2020)
12. de Vos, M., Pouwelse, J.: Real-time money routing by trusting strangers with your funds. In: 2018 IFIP Networking Conference (IFIP Networking) and Workshops, pp. 1–9. IEEE (2018)
13. del Castillo, M.: Blockchain 50: Billion dollar babies. https://www.forbes.com/sites/michaeldelcastillo/2019/04/16/blockchain-50-billion-dollar-babies (2019)
14. Delgado-Segura, S., Pérez-Solà, C., Navarro-Arribas, G., Herrera-joancomartí, J.: A fair protocol for data trading based on bitcoin transactions. Futur. Gener. Comput. Syst. **107**, 832–840 (2020)
15. Dilley, J., Poelstra, A., Wilkins, J., Piekarska, M., Gorlick, B., Friedenbach, M.: Strong federations: An interoperable blockchain solution to centralized third-party risks. arXiv:1612.05491 (2016)
16. Douceur, J.R.: The Sybil attack. In: International Workshop on Peer-To-Peer Systems, pp. 251–260. Springer (2002)
17. Dziembowski, S., et al.: Fairswap: How to Fairly Exchange Digital Goods. In: SIGSAC, pp. 967–984. ACM (2018)
18. Eckey, L., Faust, S., Schlosser, B.: Optiswap: Fast optimistic fair exchange. In: Proceedings of the 15th ACM Asia Conference on Computer and Communications Security, pp. 543–557 (2020)
19. Feldman, M., Papadimitriou, C., Chuang, J., Stoica, I.: Free-riding and whitewashing in peer-to-peer systems. IEEE J. Select. Areas Commun. **24**(5), 1010–1019 (2006)
20. Ganne, E.: Can Blockchain revolutionize international trade? (2018)
21. Gilchrist, A.: Industry 4.0: The Industrial Internet of Things. Springer, New York (2016)
22. Haeberlen, A. et al.: Peerreview: Practical accountability for distributed systems. ACM SIGOPS Oper. Syst. Rev. **41**(6), 175–188 (2007)
23. Han, R., Lin, H., Yu, J.: On the optionality and fairness of atomic swaps. In: Proceedings of the 1st ACM Conference on Advances in Financial Technologies, pp. 62–75. ACM (2019)
24. Heilman, E., Lipmann, S., Goldberg, S.: The Arwen trading protocols. In: International Conference on Financial Cryptography and Data Security, pp. 156–173. Springer (2020)
25. Herlihy, M.: Atomic cross-chain swaps. In: Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, pp. 245–254. ACM (2018)
26. Hewett, N., Lehmacher, W., Wang, Y.: Inclusive Deployment of Blockchain for Supply Chains. World Economic Forum, Cologny (2019)
27. Hyperledger: Hyperledger quilt. https://github.com/hyperledger/quilt (2020)
28. Kailar, R.: Accountability in electronic commerce protocols. IEEE Trans. Softw. Eng. **22**(5), 313–328 (1996)
29. Khalil, R., Gervais, A., Felley, G.: Tex-a securely scalable trustless exchange. IACR Cryptol. ePrint Arch. **2019**, 265 (2019)
30. Koens, T., Poll, E.: Assessing interoperability solutions for distributed ledgers. Pervasive Mob. Comput. **59**, 101079 (2019)
31. Kwon, J.: Tendermint: Consensus without mining. Draft v. 0.6 fall 1(11) (2014)
32. Kwon, J., Buchman, E.: Cosmos - A network of distributed ledgers. https://cosmos.network/cosmos-whitepaper.pdf (2014)
33. Li, D., Liu, J., Tang, Z., Wu, Q., Guan, Z.: Agentchain: A decentralized cross-chain exchange system. In: 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering (Trustcom/BigdataSE), pp. 491–498. IEEE (2019)
34. Maymounkov, P., Mazieres, D.: Kademlia: A peer-to-peer information system based on the Xor metric. In: International Workshop on Peer-To-Peer Systems, pp. 53–65. Springer (2002)

35. McConaghy, T., Marques, R., Müller, A., De Jonghe, D., McConaghy, T., McMullen, G., Henderson, R., Bellemare, S., Granzotto, A.: Bigchaindb: A scalable blockchain database. white paper bigchainDB (2016)
36. Miller, A., Xia, Y., Croman, K., Shi, E., Song, D.: The honey badger of bft protocols. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 31–42 (2016)
37. Mogan, J.: Quorum. advancing blockchain technology. En línia. Available: https://www.jpmorgan.com/country/US/EN/Quorum (2018)
38. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. http://www.bitcoin.org/bitcoin.pdf (2009)
39. Nikander, P., Autiosalo, J., Paavolainen, S.: Interledger for the industrial internet of things. In: 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), vol. 1, pp. 908–915. IEEE (2019)
40. Nolan, T.: Atomic swaps using cut and choose (2016)
41. Otte, P., de Vos, M., Pouwelse, J.: Trustchain: A sybil-resistant scalable blockchain. Future Gener. Comput. Syst. https://doi.org/10.1016/j.future.2017.08.048 (2017)
42. Oved, M., Mosites, D.: Swap: A peer-to-peer for trading ethereum tokens (2017)
43. Peters, G.W., Panayi, E.: Understanding modern banking ledgers through blockchain technologies: Future of transaction processing and smart contracts on the internet of money. In: Banking Beyond Banks and Money, pp. 239–278. Springer (2016)
44. Project, I.: Interledger protocol v4. https://interledger.org/rfcs/0027-interledger-protocol-4/ (2018)
45. Ray, I., Ray, I.: Fair exchange in e-commerce. ACM SIGecom Exchanges **3**(2), 9–17 (2002)
46. Schuh, F., Larimer, D.: Bitshares 2.0: Financial smart contract platform (2015)
47. Schulte, S., Sigwart, M., Frauenthaler, P., Borkowski, M.: Towards Blockchain Interoperability International Conference on Business Process Management, Pp. 3–10. Springer (2019)
48. Singh, A., Click, K., Parizi, R.M., Zhang, Q., Dehghantanha, A., Choo, K.K.R.: Sidechain technologies in blockchain networks: An examination and state-of-the-art review. J. Netw. Comput. Appl. **102471**, 149 (2020)
49. Specht, S.M., Lee, R.B.: Distributed denial of service: Taxonomies of attacks, tools, and countermeasures. In: Proceedings of the ISCA 17th International Conference on Parallel and Distributed Computing Systems, September 15-17, 2004, pp. 543–550. The Canterbury Hotel, San Francisco (2004)
50. Stokkink, Q., Pouwelse, J.: Deployment of a blockchain-based self-sovereign identity. arXiv:1806.01926 (2018)
51. Team, P.: Poa network whitepaper. https://github.com/poanetwork/wiki/wiki/POA-Network-Whitepaper (2018)
52. Team, A.: Ark ecosystem whitepaper. https://ark.io/Whitepaper.pdf (2019)
53. Thomas, S., Schwartz, E.: A protocol for interledger payments. https://interledger.org/interledger.pdf (2015)
54. Vo, H.T., Wang, Z., Karunamoorthy, D., Wagner, J., Abebe, E., Mohania, M.: Internet of Blockchains: Techniques and challenges ahead. In: 2018 IEEE International Conference on Internet of Things (Ithings) and IEEE Green Computing and Communications (Greencom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (Smartdata), pp. 1574–1581. IEEE (2018)
55. Vukolić, M.: Rethinking permissioned blockchains. In: Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts, pp. 3–7 (2017)
56. Warren, W., Bandeali, A.: 0x: an open protocol for decentralized exchange on the ethereum blockchain (2017)
57. Wood, G.: Polkadot: Vision for a heterogeneous multi-chain framework. White Paper (2016)
58. Yli-Huumo, J., Ko, D., Choi, S., Park, S., Smolander, K.: Where is current research on blockchain technology?–a systematic review. Plos One **11**(10), e0163477 (2016)
59. Zamyatin, A., Al-Bassam, M., Zindros, D., Kokoris-Kogias, E., Moreno-Sanchez, P., Kiayias, A., Knottenbelt, W.J.: Sok: Communication across distributed ledgers. Tech. rep., IACR Cryptology ePrint Archive, 2019: 1128 (2019)

Affiliations

Martijn de Vos[1] · **Can Umut Ileri**[1] · **Johan Pouwelse**[1]

Can Umut Ileri
c.u.ileri@tudelft.nl

Johan Pouwelse
j.a.pouwelse@tudelft.nl

1 Delft University of Technology, Delft, The Netherlands