Understanding the Future of Financial Technology (FinTech) Using Blockchains

Seyedehmahsa Moosavi

A Report

In

The Department

Of

Concordia Institute For Information Systems Engineering

Presented In Partial Fulfilment Of The Requirements

For The Degree Of Doctoral Philosophy

In Information And Systems Engineering At

Concordia University

Montréal, Québec, Canada

January 29, 2020

CONCORDIA UNIVERSITY

Division of Graduate Studies

This is to certify that the thesis prepared

By : **Seyedehmahsa Moosavi**

Entitled : **Understanding the Future of Financial Technology (FinTech) Using Blockchains**

and submitted in partial fulfilment of the requirements for the degree of

Doctoral of Philosophy

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee :

_____ Chair

_____ Examiner

_____ External Examiner

_____ Supervisor
Dr. Jeremy Clark

Approved by _____
Dr. Mohammad Mannan
Graduate Program Director

_____ 2020. _____
Dr. A. Asif, Dean
Gina Cody School of Engineering and Computer Science

ABSTRACT

Understanding the Future of Financial Technology (FinTech) Using Blockchains

Seyedehmahsa Moosavi

Blockchain technologies provide transparent, publicly accessible, and trust-less ledger that is open to anyone who wants to participate. Using public key encryption and proof of work methods, these technologies are robust against a range of faulty and malicious behaviours and create a common digital history that is validated by the "majority" (technically, a computational majority) of the network. Although blockchain was first introduced as an underlying technology of Bitcoin, its potential is not limited to that. The characteristics of this innovative technology (*i.e.,* immutability, transparency, decentralization, and automation) have led to several applications in various sectors including banking sectors, finance, healthcare, *etc.*

The rapid growth of the digital assets and blockchain-enabled technologies has resulted in fast growing cryptocurrency exchanges all over the world. These exchanges facilitate the trading process of digital assets. However, most of these trading systems use centralized servers or parties in their implementations to provide their functionality. In such systems, if one trusted party gets hacked or acts maliciously, the security of the entire system will be threatened. Therefore, the need for a highly secure cryptocurrency exchange and trading system is crucial. In this report, we study three major research works that explore the area of decentralized cryptocurrency exchanges.

Contents

List of Figures

1 Introduction

Blockchain is a type of distributed database (or ledger) that is open to anyone who wants to participate, robust against a wide range of faulty and malicious behaviours, and runs without anyone in charge. When a participant looks at her local copy of the ledger, she is assured that (i) anyone has the exact same records and (ii) each record was validated by the "majority" (technically, a computational majority) of participants before it was written into the ledger. Blockchain technology has rapidly gained global interest and become one of the most significant technologies in recent years. It was first introduced in 2008 by Satoshi Nakamoto as an underlying technology of Bitcoin, a peer to peer electronic cash system [1], whose currency reached a market capitalization of $137 billion as of January 2020. In 2014, a new blockchain based application known as Ethereum was introduced by Buterin [2]. By implementing a decentralized virtual machine, Ethereum virtual machine (EVM), Ethereum allows network users to execute programmable smart contracts on it. So developers are now able to build decentralized applications (DApps) that are executed correctly according to the consensus rules of the Ethereum network.

2 Cryptocurrency Exchanges

Strong interest in blockchain and distributed ledger technologies has led into the high trading volume of digital assets and emergence of fast growing cryptocurrency exchanges all over the world. These crypto exchanges (hereafter referred to as exchanges) are similar to stock exchanges but, as the name suggests, facilitate the trading process of different cryptocurrency tokens. Basically, such exchanges provide a trading venue where market participants can buy and/or sell cryptocurrency assets among one another. In this section, we describe two types of trading systems and pinpoint why the need for a highly secure exchange and trading system for cryptocurrencies is crucial.

2.1 Centralized Exchanges (CEX)

This group of exchanges operate in a traditional manner by providing a centralized platform controlled by exchange operators. Essentially, the exchange acts as a trusted party whom the market participants must trust to handle their assets. Thus, traders transfer the ownership of their assets to the exchange who safeguards customer funds using different methods. Although this group of exchanges make it easier for users to trade digital assets (as it takes the burden of safeguarding off traders), it requires traders to fully trust the exchange operator in taking custody of their assets. Like any other centralized platform, these exchanges are not completely immune to malicious activities. For example, in February 2014, a famous centralized Tokyo-based Bitcoin exchange called MtGox was hacked which led to loss of 650,000 Bitcoins [3]. Also, a malicious exchange can simply steal users' funds, in February 2019, QuadrigaCX, one of Canada's well-known centralized cryptocurrency exchanges, claimed that it cannot access to $190-million worth of customers' funds [4].

2.2 Decentralized Exchanges (DEX)

Unlike centralized exchanges, this group of exchanges do not depend on a centralized trusted party to facilitate the trading procedures for market participants. Decentralized exchanges use blockchain technologies to create a secure automated process that allows trades to happen directly between users (peer-to-peer). Obviously, traders do not have to trust the security of the exchange platform or honesty of the operators who holds the ownership of their valuable assets.

However, designing and implementing decentralized exchanges is not straightforward and has several technical challenges to overcome. In this report, we summarize and discuss three of the most recent research works that shed light on various landscapes of designing a decentralized exchange.

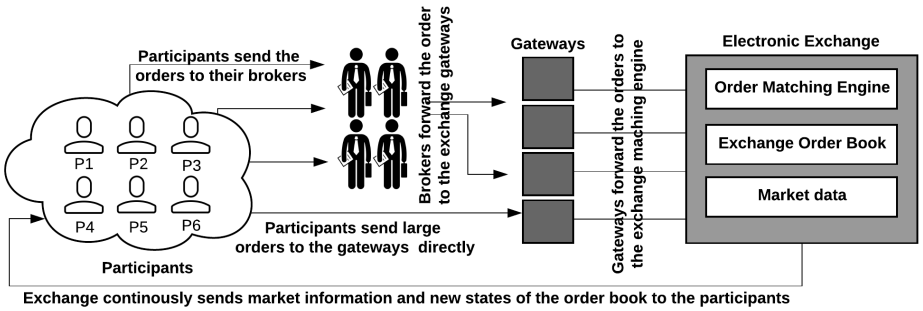

Figure 1: Representation of components and market participants.

3 Libra: Fair Order-Matching for Electronic Financial Exchanges

Unlike other research that focus on the exchange infrastructure itself, in this work, the authors discuss the importance of the *order matching policies* or simply *matching systems* used by the exchanges to match the receiving orders and execute them properly.

3.1 Market Structure

Figure 1 represents a simple electronic market where different market participants interact with one another. These participants (P_i) send their orders (*e.g.,* buy 100 shares of XYZ at \$100.00) to their brokers who then forward the orders to the market. Incoming orders are received by the exchange gateways and routed to the exchange matching engine. These gateways are located at the very short distance from the matching engine and are in several numbers to serve the load balancing purposes. The received orders are then entered to the order book[1] and sorted by the exchange matching algorithm. The orders that match will be traded against and removed from the book. Finally, the book gets updated and the new states will be disseminated to all the market participants (continuously).

Due to the limited nature of the market resources, participants need to always compete

[1]An order book is a data structure containing bid and ask orders (usually) sorted by price level

against each other in terms of *speed*. Matching policies used by the exchange decide on the order by which the order messages are placed in the book and filled. One such policy that has been widely used by electronic exchanges is price-time priority matching [5] that sorts the incoming orders on first-come-first-served (FCFS) manner. In the environment enforced by this policy, orders that are sent by faster participants have a higher chance of being placed in the book and getting filled. Oppositely, orders that are submitted by slower participants may have no chance of getting filled.

3.2 Equal Treatment in Practice

In economic theory if the exchange uses a time-sensitive order book design, the fastest market participants win in a race for a resource. The authors argue this type of fairness is difficult to achieve in practice because of the following reasons:

Jitter. An electronic exchange contains several software and hardware components that vary in their processing time from a tenths of nanoseconds to a few hundred microseconds. If the jitter introduced by exchange affects all market participants equally, it will not have a negative influence on the exchange's fairness.

Uneven Delays. As mentioned earlier, electronic exchanges often have a distributed architecture *e.g.,* load balancing servers and order gateway redundancy. Although this allows exchanges to serve the market participants more efficiently, it introduces uneven delays in the order processing times as each of these components exhibits a different performance time. For example, the order gateway used by Alice may be less crowded than the one used by Bob, in this case Alice's order gets filled before Bob's, eventhough she sends her order after.

Technical Market Manipulation. This attack, as the name suggests, is a process of exploiting the design and implementation of the electronic exchange. Malicious market participants can collude with an exchange insider who provides them with the information about the exchange *i.e.,* components' jitter and uneven delays. Such attack affected The National Stock Exchange of India where a malicious market participant managed to get

unfair privileges including (i) obtaining information about the least congested order gateways and how to access them and (ii) preventing other participants' orders from being sent to the market by keeping the gateway traffic clogged [4].

3.3 What is Fairness?

In this work, the authors define and study *fairness*– that is, what is fairness in general and what are the reasons it cannot be fully realized in financial exchanges:

Temporal Fairness. Assume that two market participants P_1 and P_2 are competing for a trading opportunity OP on a financial exchange E. With more expertise and technical capabilities, P_2 is now a faster market participant that P_1. Also, both P_1 and P_2 respond to the same stimulus from E. In this scenario, E is temporally fair, if the probability that P_1 captures the OP before P_2 does not exceed 0.5. This definition of fairness also allows the order matching policies to bring slower and faster participant into equal footing.

3.4 Reordering Algorithms

As described in 3.1, the existing asymmetries in the design and implementation of financial exchanges do not allow them to provide a fair market for participants. These discrepancies (*e.g.,* jitter and uneven delays) are not in control of the exchange operators hence they are especially difficult to combat and account for. However, exchanges can use reordering policies to enforce order equalization– the orders that arrive within a specific time margins are given equal priority. As shown in Figure 2, orders are intercepted by a software module before reaching the exchange matching engine, this component runs the reordering algorithm specified by the exchange and assigns order priority accordingly. Once incoming orders are reordered and prioritized, they are forwarded to the matching engine to be executed.

Although reordering policies seem easy to implement, they need to meet determined properties to ensure temporal fairness defined in 3.3. These properties are defined by the authors as follows:

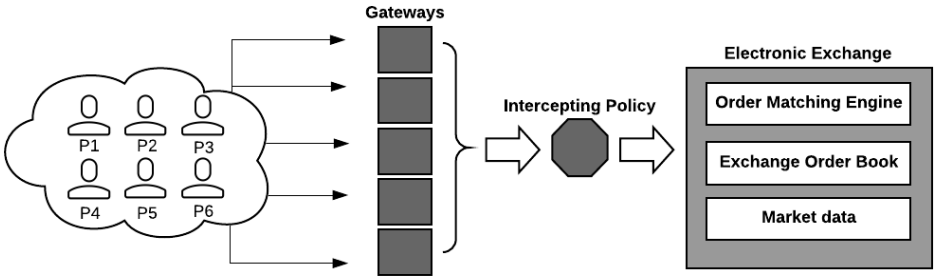

Figure 2: Intercepting component intercepts the orders with the reordering purpose and forward them to the matching engine of the exchange.

Correctness. Reordering policies should manage all the incoming order messages fairly. As discussed in 3.3, an electronic exchange E is fair if slower market participants are never in a leveraged position over faster ones.

Robustness. Reordering policies must be robust against both system malfunctions (*e.g.,* networking issues, DNS problems, server down issues) and manipulation attacks (*e.g.,* technical manipulation attacks (see 3.2)).

Minimum Impact. Exchange operators are often very careful about the long delays that could push the market participants to another operators with shorter delays. For this reason, a reordering algorithm must be designed separately from the market structure itself as any minor alteration in the algorithm may cause inconsistency in the system and generate long delays. In addition, similar to any other algorithm, these policies must be straightforward as any design complication may result in error prone implementation that is also difficult to debug.

3.5 Evaluation of Existing Reordering Algorithms

This section evaluates two order matching algorithms that are widely used by various financial exchange operators. As part of the evaluation setting, authors assume that in both cases there are two market participants P_1 and P_2 with the same reaction times. However, P_1 is unfairly advantaged by d_τ units of time over P_2 whose messages are influenced by exchange

uneven delays. They also assume d_τ never exceeds 1 millisecond.

3.5.1 First Come, First Served (FCFS)

As mentioned in 3.1, FCFS is a policy that has been widely used by Continuous Limit Order Book (CLOB) market designs [6]. This policy sorts the incoming orders on first-come-first-served (FCFS) manner. So orders that are sent by faster participants have a higher chance of being placed in the book and getting filled as opposed to those that are received later. Accordingly, FCFS policy successfully fulfills the correctness property defined earlier. However, FCFS cannot fulfill the robustness property in the existence of uneven delays. Although P_1 and P_2 submits their orders at $t = 0$, P_1's orders get executed first.

3.5.2 Frequent Batch Auctions

A different method of implementing a decentralized exchange is to use the frequent batch auction or call market design instead of a time-sensitive order book [7]. Such design can extensively address front-running issues in markets [8] hence has recently received significant attention by academics. Call market designs have two main differences with continuous limit order book: (i) time is treated as discrete, not continuous, and (ii) orders are accumulated over predetermined time intervals and are processed in batch, using a uniform-price auction, instead of serially in order of receipt. In a call market design, the probability that P_2 (who is slower by d_τ units of time) wins P_1 is:

$$\frac{L - d_\tau}{L} \times \frac{1}{2}$$

According to 3.5.2, by increasing the length of the batching interval (L), P_2's winning chance increases, however, to completely address the uneven delays between the participants (and ensure fairness) large (L) is required. This introduces a burden to exchange venues and trade-off between achieving fairness and exchange functionality.

3.6 Libra Order Matching Policy

Here we explain *Libra*; a temporally fair reordering policy that is proposed by Mavroudis*et al.* [9]. The authors first describe how this algorithm works, next they examine it against the three properties (fairness, robustness, and minimum impact) to assure this algorithm handles any type of order in a fair manner.

3.6.1 Libra's Reordering Algorithm

Libra's order matching policy considers five fields for each order-message (see Table 1) and functions in two main stages; buffering and draining. In each stage an specific algorithm is executed on the incoming order-messages as followings.

– Upon receipt of an order-message, the policy component executes $Algorithm1$ (buffering) on the order, the limit price of the order is compared to the best ask or bid in its instrument's limit order book [2]. If the order is marketable [3], it is either put in the marketable buffer $M[instrument][buy]$ or $M[instrument][sell]$ with a Null price depending on its side (buy or sell). If the order is not marketable, it is placed in its corresponding buffer, which is determined based on the instrument the order aims to trade, its side (buy or sell), and its limit price. In this case, there are a set of buffers for each instrument on the exchange venue.

– Every time a buffer receives an order that modifies its state from "empty" to "non-empty", its associated timer starts counting up to a predetermined amount of time. Once timer reaches its predetermined time, $Algorithm2$ (draining) is executed on the buffer.

– $Algorithm2$ is executed on the buffer to remove all its order-messages and forward them to be matched against its instrument's limit order book. For each participant in the buffer, it creates a list ($P[]$) and places messages from the same participant (based on the O_u identifier of each order-message) in its associated list. Each participant list is then sorted and orders are ranked in an ascending order based on their arrival time. Next, the al-

[2] In Finance, an instrument is an asset that can be traded *e.g.,* security, commodity, derivatives [10].

[3] A buy order is marketable if the buy price is greater than or equal to the Best Offer (BO). Similarly, a sell order is marketable if it is less than or equal to the Best Bid (BB) of the limit order book.

gorithm determines a random ordering of the participants and repeatedly iterates over the participants' lists accordingly. In each round, the algorithm picks one of the list, removes its top order, and sends it to the matching engine. This process continues until all of the lists are empty. For example, if the order-messages are received by the buffer in the following order $[Alice_1, Bob_1, Bob_2, Alice_2, Alice_3, Carol_1]$ and the random ordering of participant is $[Bob, Alice, Carol]$, then the lists are drained in the following sequence $[Bob_1, Alice_1, Carol_1, Bob_2, Alice_2, Alice_3]$.

Symbol	Name	Description
O_u	Identifier	Unique Market Participant Identifier
O_i	Instrument	Unique Identifier of Asset Traded
O_ut	Type	Order Types (*e.g.,* Market, Limit, Cancel, IOC)
O_s	Side	Buy or Sell Order
O_p	Price	Limit Order Price

Table 1: The orders' structure in Libra reordering policy.

3.6.2 Libra's Evaluation

Eventually, the authors evaluate Libra based on the three properties of a fair order matching policy discussed in Section 3.4.

Correctness. Libra's specifications ensure fairness under ideal, non-faulty conditions. Using this order matching policy, slow market participants are never in a leveraged position over faster ones.

Robustness. "Placeholding" is a malicious activity that can be used to exploit order matching policies. Here we show how Libra is robust against such attack. In general, market participants do not know the information content of future economic events that will cause

them to compete for resources on the exchange venue, however sometimes they may know the timing of such events (*e.g.,* regularly scheduled economic data releases). So a fast participant P_1 could maliciously submit a placeholding order before the occurrence of such event [4]. Doing so, P_1 triggers the buffer's timer early and submits her actual order quickly before the timer elapses. This will shorten the period the timer runs for after the actual event and leave P_2, a slower participant, less time to submit her legitimate order. To mitigate this unfavourable behaviour, Libra uses separate buffers for various price levels, hence P_1's placeholding order is placed in a separate buffer that P_2 will never compete for. P_1 can alternatively submit a very small placeholding order at the right price level (*e.g.,* 10 shares of XYZ). However, as in Libra orders are ranked in an ascending order, P_1's phony order is placed at the top of P_1's list and her actual order that is submitted after the economic event is executed after other participants' first orders.

Minimum Impact. Libra achieves fairness and the property of minimum impact by introducing order-triggered timers and generating a random ordering of the participants while draining the buffers. Doing so, orders that are submitted within the same time interval have equal probability of being forwarded to the matching engine first.

4 FuturesMEX: Secure, Distributed Futures Market Exchange

A futures market is a double auction market where market participants (also known as traders) can bid with both buy and sell orders. These traders can post both bid and ask orders for futures contracts– standardized promises to buy and sell some assets *e.g.,* bouches of corn and barrels of coins. These promises are made today but will be fulfilled in a future date that is specified in the contract [11]. In addition, traders deposit a certain amount of cash reserve into their margin account (initial margin) at the exchange to make sure that

[4]Placeholding order, also known as a stub quote, is a buy or sell order that is deliberately set far lower or higher than the current market price.

they can meet the promises and hence initiate their daily trading activities. All these trading activities are managed by the exchange platform. Chicago Mercantile Exchange [12] is one of these largest platforms in the world. As of today all the futures exchanges (*e.g.,* CME) are centralized. In this research work, the authors present a new distributed futures exchange called the *FuturesMex* and explain the technical challenges to overcome when designing such system.

4.1 Futures Exchange Functionalities

In order to better understand the FuturesMEX (a decentralized futures exchange) one needs to understand how a traditional futures exchange (*e.g.,* CME) works. These functionalities of a futures exchange are briefly described in the following.

Price Discovery and Maintaining the Order Book. The exchange publishes the central limit order book– electronic list of all waiting bids (buy orders) and asks (sell orders), constantly. It performs the action of aggregating all the order messages while protecting traders' anonymity (see Section 4.2 for the importance of this security property in futures markets). Figure 3 shows a limit order book where bids and asks are sorted based on their price, the exchange obtains the market price by taking the average between the best bid price and best sell price.

Order Matching. The second functionality of the exchange is to match the orders on the opposite side. Various order matching techniques exist but the most widely used is called the first-come-first-served (FCFS) (see Section 3.5.1) which sorts and executes the incoming orders on a first-come-first-served manner.

Risk Management. In order to protect market integrity and reduce risks, an exchange acts as an intermediary between the traders (both buy and sell side) and assures that all the participants honour their obligations. To do that, the exchange imposes margin (initial and maintenance) system. In such system, traders must deposit enough amount of funding into their margin account at the exchange (*the initial margin*) in order to start their trading

	Price	Volume
Sell Limit Orders	7.1	100
	6	120
	5	100 Best Offer
Market Price = (5 + 2) / 2 = 3.5		
Buy Limit Orders	2	200 Best Bid
	1.5	170
	0.5	90

Figure 3: An order book that aggregates buy and sell limit orders. The market price is an average between the best offer and the best bid price.

activities. As the market evolves, the exchange monitors every traders' margin account to make sure they have enough funding (*the maintenance margin*). Due to the market price fluctuations, traders' reserve could fall below the maintenance margin, if so, the exchange asks them to deposit extra funding into their margin account (*margin call*).

Minimizing Price Discrimination. In addition to market integrity, the exchange has to protect traders' anonymity and confidentiality. In order for traders to open trading accounts at the exchange, they must confirm their identities through some verification procedures called *Know Your Customer (KYC)* [13]. However, the exchange must not publicize any of these information (even traders' IDs) in the order book. As it is presented in Figure 3, the order book must only contain the order messages with their price and volume. In Section 4.2 we explain the consequences of braking traders' anonymity and confidentiality in a futures market.

4.2 Confidentiality and Anonymity in Futures Markets

The authors discuss that although a common belief is that confidentiality and anonymity are mostly used in preventing against snooping attacks and not in designing a futures market, these security properties must be fulfilled in FuturesMex. They argue that if these two properties fail, malicious traders can deliberately push honest traders out of the market, this is called the *price discrimination attack* which we briefly summarize in the next section to highlight the importance of confidentiality and anonymity in futures markets.

4.2.1 Price Discrimination attack

Let's assume a futures market with four market participants Alice, Bob, Carol, and Mallory. As it can be seen in top middle Figure 4, Alice holds 1400 amounts of cash in her trading account while the other three hold 1200. Alice accumulates 90 contracts at \$10 (also known as short positions) and sell 30 to each traders at the current market mid-price \$10 ((\$11+\$9)/2 = \$10). In order to fulfill her promises and liquidate her short position of 90 contracts, Alice first needs to purchase these contracts from the market at the current market price, this leads to her cash account to reduce to 500 ($1400 - 90 \times 10 = 500$). Having a limited amount of cash reserve, Alice could post a limit sell order at \$9 which could decrease the market price to \$8 if filled. Doing so, she increases her net position to 692 and gained \$192 ($1508 - 102 \times 8 = 692$). However, if anonymity and confidentiality properties are broken and Mallory knows that Alice has a limited cash reserve, she can run an evil scheme and increase the market price to more than \$10. As shown in bottom left Figure 4 by cancelling her sell order at \$11, Mallory increases the market price to \$16. Now if Alice wants to liquidate her position, she needs \$1440 ($90 \times 16 = 1440$). At this point, Alice does not have sufficient cash reserve (her net position is negative (-40)) to fulfill her promises and she is pushed out of the market.

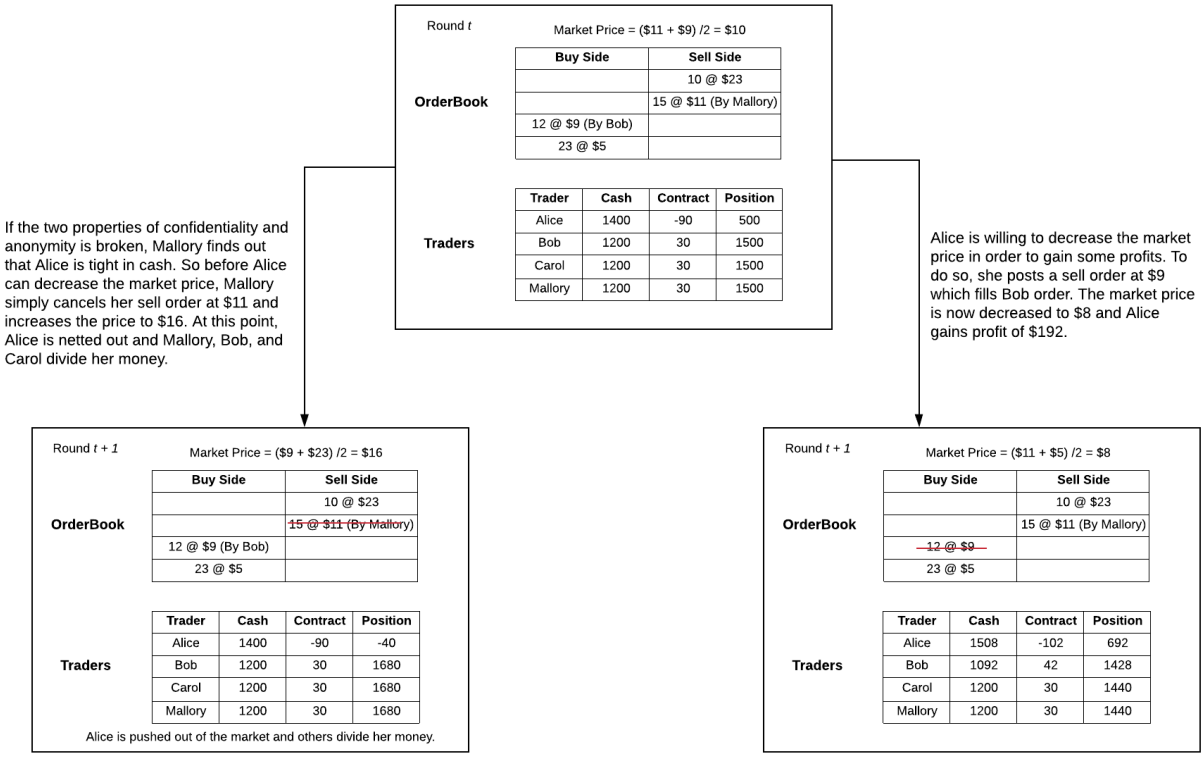

Figure 4: Price discrimination attack.

4.3 The FuturesMEX Design

FuturesMEX is a distributed futures exchange that allows traders to buy and sell promises based on limited cash reserve while enforcing the trading discipline and protecting market integrity and traders' anonymity. FuturesMEX is realized by a hybrid protocol combining standard cryptographic primitives; it uses commitment schemes and zero-knowledge proofs to provide confidentiality and integrity. It also relies on an anonymous communication channels (*e.g.,* Tor) and Merkle Tree to protect traders' anonymity. FuturesMEX overall protocol runs in four levels, which we explain at a high-level in the following.

Initialization: In order for a trader to be able to participate in a futures market, they have to prove (in secret) that they own a valid inventory *i.e.,* their position is non-negative. So, first all the traders (P_is) complete the following steps individually:

- P_i commits to its initial inventory and proves in zero knowledge that her inventory is

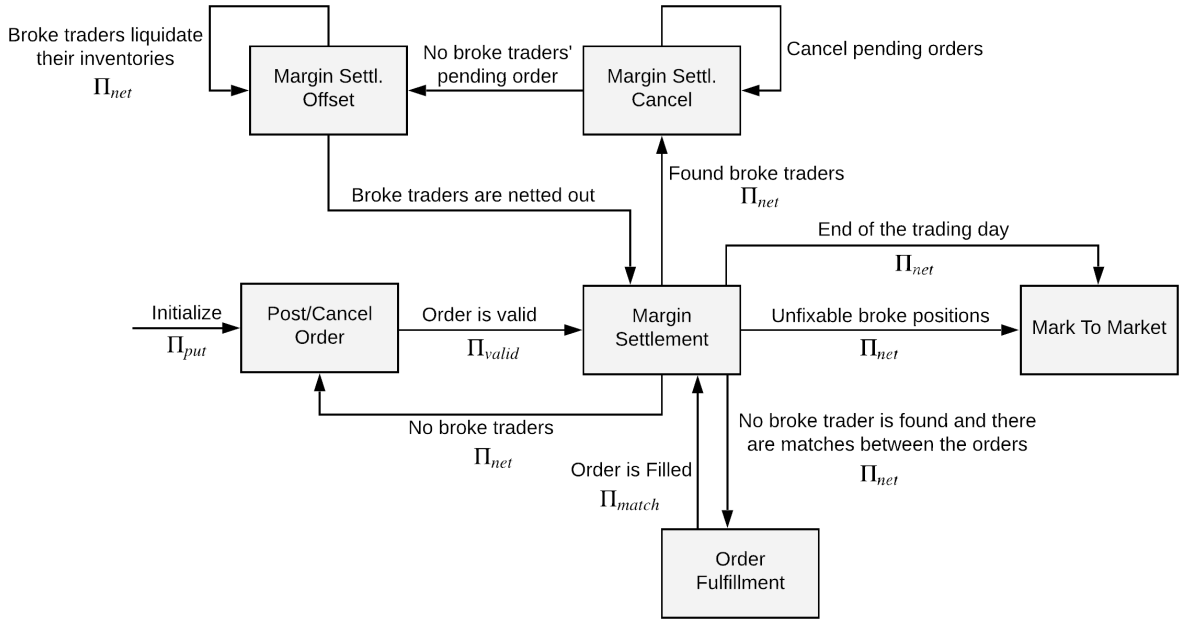

Figure 5: Futures market state transition diagram.

valid (*i.e.,* her inventory contains a non-negative cash reserve and zero volume holding). By using the commitment scheme, participants first bootstrap their private initial margin and prove that they can start their trading activities.

- By committing to an inventory, P_i acts as a prover and the other traders (as verifiers) execute the subprotocol Π_{put} which appends the commitment of P_i's inventory to a Merkle Tree τ.[5]

Post/Cancel Order: A market participant P_i can submit a new order (or cancel her previous order) once she proved that her net position is non-negative.

- All participants check whether it is a valid order or not. To do so, $P_1, ..., P_n$ run the subprotocol Π_{valid} [6] which allows P_i to perform her action if succeeds.

[5]During the execution of the various phases of the protocol, a Merkle Tree τ based on a standard collision resistant hash function, with the leaves as commitments of traders' inventories, is constructed and constantly updated. FuturesMEX protects the anonymity of traders' inventories by using the Merkle Tree τ together with zero-knowledge proofs.

[6]Traders perform this using the two functions; *get* and *put*. By executing the *get* function, they all retrieve the order's associated inventory from the Merkle Tree. By decimating to the inventory, P_i proves her net position is non-negative and that she can perform trading activities. Traders then execute the function *put*

- Posting buy and sell orders in a futures exchange could effectively lead to changes in the the market mid-price. Market price fluctuations immediately affect the instant net positions of other traders (and in some cases push participants with negative net positions out of the market) [14]. To prevent this, every time (i) a post/cancel is added to the order book or (ii) two orders are executed between two traders, the protocol checks all the participants' inventory for negative net position. In this step, all traders run the subprotocol Π_{net} which checks everyone's net positions, if any negative net position is found, the new update is reversed and the protocol proceeds to the **Margin Settlement** phase.
- All participants run the Π_{match} subprotocol, which matches the orders against each other.
- Since the new updates in the order book could change traders' net positions, all the traders run the subprotocol Π_{net} again (after the match happens).

Margin Settlement: This phase is (re)visited every time there is a trader P_i in the system whose net position is negative and she has insufficient funds in her margin account (also known as a broke trader). Once the following steps are completed, the protocol resumes its previous phase (goes back to where it was). As it can be seen in Figure 5, this phase is grouped into three sub-phases: (i) Margin Settl., (ii) Margin Settl. Cancel, and (iii) Margin Settl. Offset.

- Margin Settl.: All traders' net positions are checked and evaluated to identify the broke traders.
- Margin Settl. Cancel: Every broke trader must cancel all their pending orders.
- Margin Settl. Offset: All the contracts held by the broke traders are sold/bought as market orders [7](also known as offsetting traders' positions).
- If the broke traders hold more contracts than currently available in the market or their net positions cannot go back to non-negative, the protocol goes to **Mark to Market**

to append the new inventory to the Merkle Tree.

[7]Market orders are executed at the best available price in the current market.

phase.

- Once there is no broke traders in the system and there are matches in the order book, the protocol goes to the **Order Fulfillment** phase.

Mark to Market: This phase is executed at the end of the trading day or during the **Margin Settlement**. In this step, all the traders (whose net positions are non-negative) sell/buy all their holding contracts (offset their positions) at whatever price is available in the market, also known as market orders.

5 Tesseract: Real-Time Cryptocurrency Exchange Using Trusted Hardware

As mentioned in 2.1, for traders to be able to trade on centralized exchange services, they must transfer their funds into the exchange's centralized servers through the web applications. However, placing too much trust on these services can lead to loss of funds. When trading on decentralized exchanges, traders can trade digital assets while maintaining the ownership of those assets. Although these services are designed to avoid centralization and eliminate the trust assumptions, they do not allow the market participants to trade in real-time hence they cannot react to the market price fluctuations. In this research work, the authors propose a secure decentralized exchange service called *Tesseract* that allows traders to swap their assets atomically in a real-time.

5.1 The Tesseract Design

Tesseract exchange [15] achieves its security and performance goals using Intel SGX. The hash of the Bitcoin genesis block is encoded into the enclave. As the execution begins, the enclave fetches the latest Bitcoin block headers from an untrusted Bitcoin client and validates each header's proof-of-work (PoW) with respect to the current difficulty of the Bitcoin protocol. Once validated successfully, the enclave adds the block headers to a queue that

it maintains. Note that there is a similar queue inside the enclave for every cryptocurrency that the Tesseract exchange supports. Once the queue is initialized, the enclave generates a keypair (sk, pk) for every supported cryptocurrency, it then publishes a Quote attesting to the pk as its deposit address for each cryptocurrency. To achieve a higher level of security, these keys are re-generated after each settlement transaction. As shown in Figure 6, to open a Tesseract account, Alice creates a time-locked transaction [8] that deposits an amount of cryptocurrency into the corresponding deposit address of the Tesseract exchange. Alice then provides the enclave with the authentication path of the deposit transaction inside the Merkle Tree of the block that contains the transaction. To validate the deposit transaction, the enclave checks if (i) the block's header is included in the queue it maintains and that (ii) additional amount of blocks are built on top of that block. Once validated, Alice is credited a deposit account inside the enclave that contains the balance she transferred to Tesseract. At this point, Alice can start trading on Tesseract in real-time by sending bids and asks order messages to the Tesseract server using a secure channel. The anonymized version of the order book (where usernames are hidden) is publicized by the Tesseract server together with a quote to attest to the current state of the book. Recurring trades on the exchange constantly modify participants' balances inside the enclave. So instead of reflecting these account modifications into the blockchain, Tesseract allows trades to happen inside the enclave and at the end of the trading day, it broadcasts a settlement transaction to the associated blockchain. In order to pay the settlement transaction fees, Tesseract collects proportional fees for every trade that is successfully executed. As mentioned, users bids and asks order messages are sent to the Tesseract server through mutually authenticated TLS sessions. Doing so, Tesseract is secure against front-running attacks as an adversary, who monitors the traffics, cannot gain any information about the orders.

[8] Such transaction specifies a time limit (*e.g.,* three weeks). Before that, the enclave is the only party who can spend the deposit amount. Once the limit is reached, Alice takes the control of the deposit amount back.

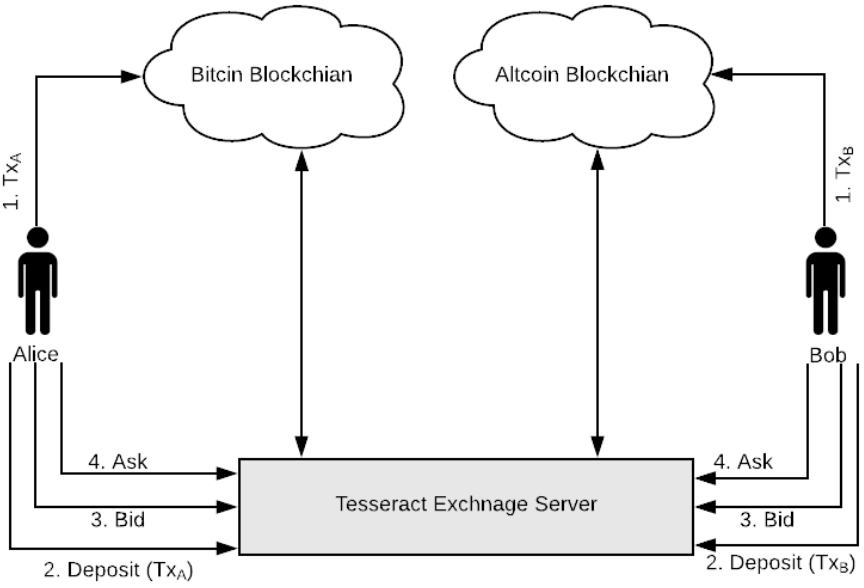

Figure 6: Representation of Tesseract's account initialization.

5.2 Atomic Cross-Chain Trading

Any decentralized exchange must facilitate trading processes to happen atomically– atomic swap is a process of trading a cryptocurrency for another one without trusting each other. If atomic cross-chain fails, a malicious trader can steal funds from the other use who stayed honest through the trading process. Consider a simple scenario where two traders Alice and Bob open Tesseract accounts and are ready to trade. Alice owns 5 Bitcoins which she wants to trade with 100 Altcoins. On the other side, Bob owns 100 Altcoins and is ready to trade these with 5 Bitcoins. Both Alice and Bob send order messages to the order book and the trade happens. At the end of the trading day, the enclave constructs two settlement transactions T_1 and T_2 which reflect the account balances on Bitcoin and Altcoin networks respectively. In such scenario, a malicious adversary who has physical access to Tesseract server and is monitoring the network traffic can collude with Alice and intercept both transactions before they reach to the network. The adversary holds T_1 and sends T_2 to the Altcoin network. Once T_2 is confirmed on the Altcoin network, Bob's 100 Altcoins are transferred to Alice,

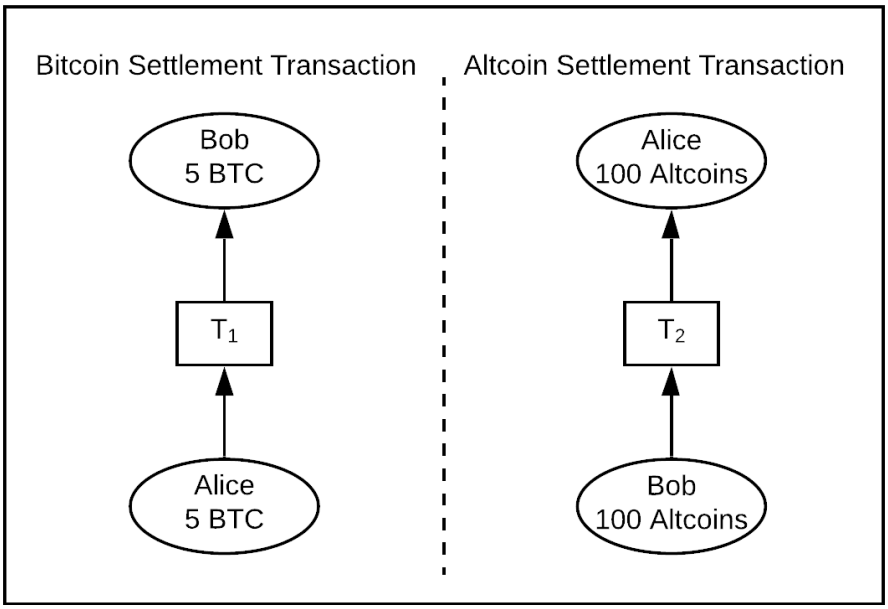

Figure 7: Representation of cross-chain trading problem.

while Alice still owns her 5 Bitcoins (see Figure 7). To prevent this, Tesseract uses an intricate solution which enforces the settlement transactions to either happen completely and atomically or not at all. This solution is called *all-or-nothing settlement* and is defined as follows. When two settlement transactions T_A and T_B are sent to cryptocurrency networks A and B:

1. Both T_A and T_B are confirmed on their cryptocurrency networks A and B.
2. Neither T_A and T_B are confirmed on their cryptocurrency networks A and B.

5.3 Atomic Swap: Naive Protocol

To pinpoint the reason why the intricate protocol is needed to address all-or-nothing settlement problem between N market participants, the authors first present their naive protocol Π_{simple} which relies on N reputable servers $S_1, S_2, ..., S_N$. To assure all-or-nothing settlement in the example in Section 5.2, protocol Π_{simple} works as following:

1. The enclave first generates a symmetric key K that will be used for symmetric encryp-

tion.

2. The enclave inserts K into T_1 and T_2. Note that no malicious party can extract K from T_1 and T_2 as the signatures for these transactions are over the entire transaction data.
3. The enclave forwards the ciphertext $ct = encrypt_k(T_1, T_2)$ to $S_1, ..., S_N$ and waits for acknowledgement of receipts.
4. The enclave broadcasts T_1 and T_2 to Bitcoin and Altcoin blockchain networks respectively.
5. Each server (S_i) that sees T_2 but not T_1, fetches K from T_2, decrypts the ciphertext it has in its possession and broadcasts T_1 to its blockchain (Bitcoin in this example).

However, the protocol Π_{simple} is yet vulnerable to the race condition attack. A malicious adversary, who monitors the network traffic, can intercept T_1 and T_2 and does not forward them to their blockchain networks. As mentioned in Section 5.1, deposit transactions are time-locked. The adversary holds the transitions until T_1's time-lock (that belongs to a corrupt user Alice) is about to expire. Once the time limit is reached, Alice is instructed to redeem her deposit. The adversary then broadcasts T_2 to the Altcoin network, an honest server (S_i) that sees T_2 but not T_1, will broadcast the T_1 to the Bitcoin blockchain, however, this transaction will not go through because it tries to spend an output which is already spent by Alice earlier.

5.4 Atomic Swap: Practical Protocol

As explained in the previous section, the settlement protocol Π_{simple} is vulnerable to the race condition attack. In order to address this problem, the authors introduce a practical settlement protocol Π_{prac} that achieves its security by distributing trust between N servers $S_1, S_2, ..., S_N$ that run SGX enclaves. This settlement protocol is secure if there is only one server S_i that is not reachable by the adversary. In a nutshell, this protocol requires a proof that the settlement transaction T_1 is either committed to its blockchain or cancelled, and then performs the same action on T_2 correspondingly. Similar to the initialization phase of

the protocol Π_{simple}, the Tesseract server and $S_1, S_2, ..., S_N$ servers must share a symmetric key that is only reachable by their enclaves. The two transactions T_1^c and T_2^c introduced here are the cancellation transactions of T_1 and T_2 respectively [9]. The practical settlement protocol Π_{prac} for the example used in Section 5.2 works as following:

1. Tesseract forwards the ciphertext $ct = encrypt_k(T_1, T_2, T_1^c, T_2^c)$ to $S_1, ..., S_N$ and waits for acknowledgement of receipts.
2. Tesseract broadcasts T_1 to Bitcoin network.
3. Servers $S_1, S_2, ..., S_N$ monitor the Bitcoin blockchain for the confirmation of the T_1, if servers do not find T_1 confirmation within T blocks, they will broadcast the transaction T_1^c to the Bitcoin network.
4. If servers find T_1 confirmation on the Bitcoin blockchain, they will broadcast T_2 to the Altcoin blockchain (note that the servers must keep T_2 confidential inside their enclaves until the T_1 is confirmed on its blockchain).
5. If servers find T_1^c confirmation on the Bitcoin blockchain, they will broadcast T_2^c to the Altcoin blockchain.

The authors describe the following attack that can be performed against Π_{prac} by the adversary who has physical access to the Tesseract server and at least one of the trusted servers among $S_1, ..., S_N$:

1. The adversary intercepts the T_1 after it is broadcast by the Tesseract server, he then deactivates the Tesseract server.
2. The adversary generates a fake chain of blocks that contains T_1 and delivers it to the S_i.
3. Once S_i believes the T_1 is confirmed on the Bitcoin blockchain, it releases the T_2, which was kept confidential.
4. In the meantime other honest servers broadcast T_1^c to the Bitcoin network as the confirmation of T_1 is not found on the blockchain.

[9] In Bitcoin, one can construct T_1^c by simply spending one of the T_1's inputs into a new output that is similar to that input.

5. The adversary waits for T_1^c to be confirmed on the Bitcoin blockchain, and then he broadcasts T_2 to the Altcoin network.

Doing so, Bob's 100 Altcoins are transferred to Alice, while Alice maintains the ownership of her 5 Bitcoins. However in order for an adversary to be able to perform this attack, he needs to have more than half of the computational power of the blockchain network. If he does not, he cannot generate the fake chain as fast as other nodes in the network, who are building the legitimate chain, hence cannot feed the fake chain to the S_i within T_1 blocks. Therefore, this protocol is secure against such race attack.

6 Future Works

Today several research focus on decentralizing the trading procedures of cryptocurrencies. Blockchain technologies have shown so much potential to offer to different industries (*e.g.,* financial sectors, banking systems, healthcare *etc.*). However, based on the way they work, network miners and nodes are in the privileged position which itself introduces technical challenges in implementing a decentralized exchange and order book. A market participant must broadcast her order message to the network before it can be placed in the block (also known as order book). In such scenario, the privileged nodes and miners can front-run the transaction or completely drop (censor) it. Also, there is no notion of time or enforcing the priority of orders on the blockchains; if Alice is more well-connected to the network, her order gets broadcast better and executed first although she sends it after Bob. To address these challenges, my research focuses on a new solution; implementing an alternative data structure called a *call market*. Call market designs have two main differences with continuous limit order book: (i) time is treated as discrete, not continuous, and (ii) orders are accumulated over predetermined time intervals and are processed in batch, using a uniform-price auction, instead of serially in order of receipt.

The main objective in my doctoral research is to investigate and illustrate financial use

cases of the blockchain technologies. Specifically, I aim to (i) understand the landscape of options for designing a decentralized order book for exchanging digital assets that is run over an open network of computers without anyone in charge. Once I successfully design a decentralized order book with all the requirements and specifications, I aim to (ii) implement and deploy a proof of concept (on the Ethereum blockchain). I will submit the result of this research work to the *Financial Cryptography and Data Security 2021*.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," tech. rep., Manubot, 2019.

[2] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, p. 37, 2014.

[3] A. Norry, "The history of the mt gox hack: Bitcoin's biggest heist." `https://blockonomi.com/mt-gox-hack/`, June 2019. (Accessed on 12/31/2019).

[4] Securities and E. B. of India, "Sebi — order in the matter of nse colocation." `https://www.sebi.gov.in/enforcement/orders/apr-2019/order-in-the-matter-of-nse-colocation_42880.html`, 2019. (Accessed on 11/11/2019).

[5] T. Preis, "Price-time priority and pro rata matching in an order book model of financial markets," in *Econophysics of Order-driven Markets*, pp. 65–72, Springer, 2011.

[6] E. Budish, P. Cramton, and J. Shim, "The high-frequency trading arms race: Frequent batch auctions as a market design response," *The Quarterly Journal of Economics*, vol. 130, no. 4, pp. 1547–1621, 2015.

[7] J. Clark, J. Bonneau, E. W. Felten, J. A. Kroll, A. Miller, and A. Narayanan, "On decentralizing prediction markets and order books," in *Workshop on the Economics of Information Security, State College, Pennsylvania*, 2014.

[8] S. Eskandari, S. Moosavi, and J. Clark, "Sok: Transparent dishonesty: front-running attacks on blockchain," 2019.

[9] V. Mavroudis and H. Melton, "Libra: Fair order-matching for electronic financial exchanges," *arXiv preprint arXiv:1910.00321*, 2019.

[10] Investopedia, "Instrument definition." `https://www.investopedia.com/terms/i/instrument.asp`, August 2019. (Accessed on 11/25/2019).

[11] J. CHEN, "Futures contract." `https://www.investopedia.com/terms/f/futurescontract.asp`, April 2019. (Accessed on 12/09/2019).

[12] Wikipedia, "Chicago mercantile exchange - wikipedia." `https://en.wikipedia.org/wiki/Chicago_Mercantile_Exchange`, December 2019. (Accessed on 12/11/2019).

[13] Wikipedia, "Know your customer - wikipedia." `https://en.wikipedia.org/wiki/Know_your_customer`, December 2019. (Accessed on 12/11/2019).

[14] F. Massacci, C. N. Ngo, D. Venturi, and J. Williams, "Non-monotonic security protocols and failures in financial intermediation," in *Cambridge International Workshop on Security Protocols*, pp. 45–54, Springer, 2018.

[15] I. Bentov, Y. Ji, F. Zhang, Y. Li, X. Zhao, L. Breidenbach, P. Daian, and A. Juels, "Tesseract: Real-time cryptocurrency exchange using trusted hardware.," *IACR Cryptology ePrint Archive*, vol. 2017, p. 1153, 2017.