This is a timely piece that details an important problem in the application of blockchain technology for trading platforms. Most of the current applications focus on the custodial role of blockchain for equity and bond transactions/ownership. Blockchain has also been employed for transaction contracts. In the latter application, it is crucial that only the parties to the transaction should be able to see the transaction - not everyone. JP Morgan's Quorum has solved this problem by having the blockchain show only encrypted transactions. The decryption key is encrypted with the parties' public keys. Hence, only the parties to the transaction can decrypt. This is a somewhat different application that requires an Ethereum framework.

Your idea is simple. A hash of the transaction goes into a blockchain to establish time. The transactions are then ordered by the pre-established time.

Here are a few issues I would like you to consider.

1. It is not clear what this blockchain looks like and how it operates. It is populated by hashes (which establish the order) and plaintext transactions. I would like a bit more detail (indeed might be a useful appendix).

a) For example, the hashes are sent to the blockchain. How are these hashes "verified"? You can imagine an attacker flooding the network with billions of hashes.

b) Are the hashes loaded in in real time -- or are they blocked?

c) If there is verification, who does the verification? What are the mechanics?

d) For the plaintext transaction, who does the verification?

e) Is this a private blockchain where you need to be permissioned to add?

f) Can you talk more about the latency between the hash and the actual transaction?

2. I am worried about the following. What prevents a trader from constantly flooding the blockchain with hashes for different order terms, i.e. 90, 89, 88, ... Given the latency, the trader only picks the one that best suits them and reveals the plaintext order.

3. You should probably look at Quorum. There are some good ideas there that are potentially applicable. My main worry is the flooding of hashes either by strategic traders or attackers.

4. Any additional discussion about other potential problems this solves would be valuable. For example, does this technology solve spoofing issues? Is the change in latency an advantage over current systems?

**Footprints on a Blockchain:**

**Trading and Information Leakage in Distributed Ledgers**

**March 2017**

This paper examines information leakage when trading in distributed ledgers. We show how the lack of time priority in the period between the publication of a transaction and its validation by miners or designated participants can expose a transaction’s “footprint” to the market, resulting in potential front-running and manipulation. We propose a cryptographic approach for solving information leakage problems in distributed ledgers that relies on using a hash (or “fingerprint”) to secure time-priority followed by a second communication revealing more features of the underlying market transaction – in effect using a transaction’s fingerprint to hide its footprint. Solving the information leakage problem greatly expands the potential applications of private distributed ledger technology to include trading.

**Footprints on a Blockchain:**

**Trading and Information Leakage in Distributed Ledgers**

Advocates of blockchain technology see a promising future for the innovation and its application to financial back-office infrastructure. Apart from its role in cryptocurrencies like Bitcoin, distributed ledger technologies are being developed to handle stock trades, clear and settle leveraged loans, handle catastrophe reinsurance, track land titles and transactions, record the ownership of intellectual properties, facilitate peer-to-peer marketplaces – the list goes on and on. Much of the appeal of the blockchain lies in its use of decentralized distributed ledger technology in which an immutable global record emerges of the various transactions performed on the network. The ability to clear and settle assets in such a setting, along with the rise of new “smart contracts” that can automatically trigger additional steps such as recording ownership changes or authorizing payments, can have, in the words of the Bank of England, “far-reaching implications for the financial industry”.[[1]](#footnote-1)

While not disputing this optimistic outlook, we argue in this paper that this new technology comes with some very old problems. In particular, in traditional market settings a fundamental problem facing traders is how to hide their trades and trading intentions from other traders. Disguising a trade’s “footprint” is necessary to keep others from copying, or even worse, “front-running” your trades, thereby extracting rents that should have been yours. A variety of algorithmic and technological approaches focus on this task in current markets. As we discuss here, the same information leakage can take place in a distributed ledger setting. This problem arises because even though the transactions in a blockchain are time-stamped, there is no actual time priority in the process of getting a transaction validated and onto the distributed ledger. One contribution of our paper is to show how information leakage can arise in the interim period between the publication of such a transaction and its validation by miners or designated participants, exposing other participants in a distributed ledger to potential front-running and manipulation.

A second, and more important, contribution of this paper is to suggest a solution. We propose a cryptographic approach for solving information leakage problems in trading on distributed ledgers.[[2]](#footnote-2) Our two-part process uses a cryptographic hash (essentially a type of digital “fingerprint”) to secure time-priority which is then followed by a second communication revealing more features of the underlying market transaction. Because the initial hash does not reveal details of the trade, the ability to front-run a transaction before it is stored in the distributed ledger is eliminated. Using a transaction’s “fingerprint” to hide its “footprint” removes the ability to manipulate the order of transactions and so restores time priority to the validation process.

Solving the information leakage problem in distributed ledgers greatly expands the feasibility of distributed ledger applications. To date, proposed uses of distributed ledgers focus mainly on post-trade clearing and settlement where there is little ability to profit from any information leakage. In market settings, however, where information leakage is important, our analysis shows how distributed ledgers could be used to operate decentralized markets that safeguard participants’ information. As perhaps befits a new technology, we argue that modern computer science-based tools can be used to solve an old problem in a new setting.

**Information Leakage in Distributed Ledgers**

To understand why information leakage can occur, it is useful to set out the distinction between centralized and decentralized systems, and explain how their differences affect transactions processing. Centralized systems (for example, exchanges) are straightforward: they have a single trusted party, usually in a single location, processing transactions and assigning timestamps to them. Decentralized systems (for example, the Bitcoin blockchain) are much more complex: there is no trusted party, and instead it relies on a network of untrusted, independent parties executing an algorithm to process transactions.[[3]](#footnote-3) The most relevant part of this process for the focus of our paper is the sequencing of transactions, where each transaction is assigned a timestamp and an order relative to every other transaction. As we will discuss, the limitations of this sequencing process set the stage for information leakage to occur.

*Transaction processing in decentralized systems*

Due to transmission delays in a network, it is impossible to reach agreement on the actual order in which transactions happen in a decentralized system. Such latency issues are an all too familiar problem in the current high frequency market structure characterizing equity or futures trading, but for a number of reasons this problem is far more challenging in a distributed ledger setting. One such reason is that while the number of nodes (or exchanges) in markets is fairly small, the nodes in a distributed ledger system can number in the thousands, or even tens of thousands. A second is that these nodes are distributed, making transmission delays an inevitable feature of the system because different nodes see transactions appear at different times.[[4]](#footnote-4) Finally, in a decentralized system there is no central party trusted to order the transactions.

As a result, the various algorithms employed in a distributed ledger can only attempt to make all parties in the network agree on what (for short time intervals) is essentially an arbitrary ordering of the transactions.[[5]](#footnote-5) In the period between the broadcast of a Bitcoin transaction and its being processed by the network, for example, there are no guarantees offered with regards to its order relative to other transactions in the same state or to what timestamp the transaction will eventually be assigned.

Crucially, the content of any transaction (where we are using “transaction” in a broader sense to include events such as a trade, a quote, or indication of interest) that will be posted to the distributed ledger is observable by all or parts of the network in this period.[[6]](#footnote-6) This visibility allows other participants to act upon the information carried in the transaction before it has been processed and received a guaranteed ordering relative to other transactions. Depending on the algorithm employed, some participants may also be able to alter the relative ordering of transactions, arbitrarily and potentially to their own benefit. And, depending on the extent of these alterations, any changes can range from being undiscoverable to the rest of the network, to being probable but unprovable. Once the transaction is processed by the network, it will be permanently stored and given a definitive place in the order sequence in the distributed ledger. Only at this point is it possible to offer the guarantee that no new transactions end up with an earlier time priority than the original transaction.

The lack of time priority in the validation stage applies generally across distributed ledger settings, however the validation process employed in such systems may differ. As we discuss below, public distributed ledgers use miners and private distributed ledgers use an elected leader to validate transactions, but regardless of the method used, the basic problem is the same: one party is temporarily trusted with ordering the transactions because there is no global agreement on a natural ordering of transactions.

*Transaction processing in Bitcoin and Ethereum*

In a public distributed ledger system like the Bitcoin and Ethereum networks, transactions are processed by “miners” (i.e. high powered computers) that compile these transactions into chunks (known as blocks), and collect the fees associated with each transaction. There are numerous miners on the network, and which one writes any given block is unknown in advance, with the probability of doing so equal to their share of mining power relative to the total. When a transaction is broadcast, it is forwarded by the network and kept in a transaction pool known as a mempool. When a miner mines a new block, it will consist of transactions from this mempool. The miner can reorder or censor transactions as they see fit, to the point where empty blocks (with no transactions) are perfectly valid. This behavior, combined with infrequent generation of blocks, can result in unpredictable delays between a transaction being broadcast and it being assigned an order relative to other transactions in the distributed ledger.[[7]](#footnote-7) As Figure 1 shows, these delays in Bitcoin have typically been around 10 minutes, but they can be much longer due to there being no theoretical upper bound on how long it can take.

Most miners are agnostic to the content or source of transactions, and they will seek to optimize their own returns by including as many transactions as possible in each block (thereby collecting transaction fees). This category of miner will not have any incentive to manipulate the relative order of transactions, but neither will they have any interest in keeping transactions in any particular order. Thus, transactions containing conflicting orders may be shuffled arbitrarily for a time-period of multiple minutes.

Other miners, however, may have motives beyond simply collecting transaction and block-fees. For example, they can decide to inspect the orders awaiting processing in the mempool, and then put in their own conflicting orders in front of the other orders. The fact that they can blame the resulting sequencing on network delays, which do occur in decentralized systems, makes this a tricky problem to fight. That most miners are anonymous only contributes to the difficulty of knowing whether such data snooping (and front-running) has actually occurred. What is clear is that there should be no expectation of privacy for broadcast data in a public distributed ledger.

*Transaction processing in leader-driven private decentralized systems*

In contrast to the public distributed ledgers described above, there are also private distributed ledger systems. In such systems (examples include Symbiont Assembly and Hyperledger Fabric), an elected leader is responsible for processing transactions for a given period of time. These leader-driven systems do not necessarily operate on blocks, but the transactions are similarly written to a distributed ledger. Relative to Bitcoin, the delay between a transaction broadcast and it being written to a private ledger is much shorter, often on the order of seconds for a global system. That delay still provides more than enough time for other participants on the network to broadcast their own transactions with conflicting orders in them.

Like Bitcoin miners, the elected leader has a great degree of freedom in how to order transactions relative to each other. Indeed, the problem is exacerbated by the fact that the leader in this kind of system knows ahead of time that it is the one responsible for writing transactions to the ledger, and can plan accordingly. By the rules of such systems, only when a transaction is outright censored or delayed for a long period of time (several seconds) does a leader-election algorithm kick in and change the system over to a different leader. As in Bitcoin, any leader has no obligation to order transactions in the order it received them, and network delays can make the perceived order different for different observers, making it impossible to prove suspected willful reordering.

### *D. Distributed systems and time*

The issues outlined above are due to two features of distributed ledger systems: 1) they are decentralized, so there is no central authority that can force miners or leaders respectively to process transactions in any particular order, and 2) they are distributed, and due to network delays the geographically disbursed nodes may receive transactions in a different order. Systems that are distributed, but have a central authority managing them, can achieve much better performance. But even this category of distributed ledger systems will be unable to preserve correct ordering globally for transactions happening near each other in time. This lack of time priority sets the stage for information leakage and potential front-running in any distributed ledger.

**A Proposed Solution**

The basic problem is straightforward: remove the ability of miners or leaders to profit from trading on the information in your transaction. Once your transaction is written to the distributed ledger, the problem is largely ameliorated, but it is in the preceding processing stage when miners or leaders can inspect the transactions they are supposed to be ordering that information leakage can occur. So a natural starting point for a solution is to consider changing how this initial processing takes place, with the aim of establishing strict time priority for transactions. [[8]](#footnote-8)

Our proposal as applied to a financial market involves two steps. First, the details of an order are run through a cryptographic hash function (discussed in detail below), producing an identifier mathematically linked to the order, but not exposing any details of the order itself, other than the fact that some trade is desired.[[9]](#footnote-9) Broadcasting that hash value and waiting for it to be processed by the system will reserve an order’s time priority. Once the priority is secured, the order itself is broadcast. At this stage, the information leakage is not of any concern as all involved in the market will recognize the relationship between the order and its corresponding hash value, thus executing them at the reserved time priority. We now turn to specifying this process in more detail at a heuristic level, with greater explanation given in the Appendix.

*Cryptographic hashes - a transaction’s fingerprint*

A cryptographic hash function is a mathematical algorithm that performs a one-way transformation on a message consisting of arbitrary data, producing a (cryptographic) hash value.[[10]](#footnote-10) One-way means that reversing the algorithm and learning anything about the original data is infeasible. On the same note, any small change in the original data will result in an extensive change in the corresponding hash value, making it look uncorrelated with the first hash value. This property also makes it infeasible to find two different messages that produce the same hash value.

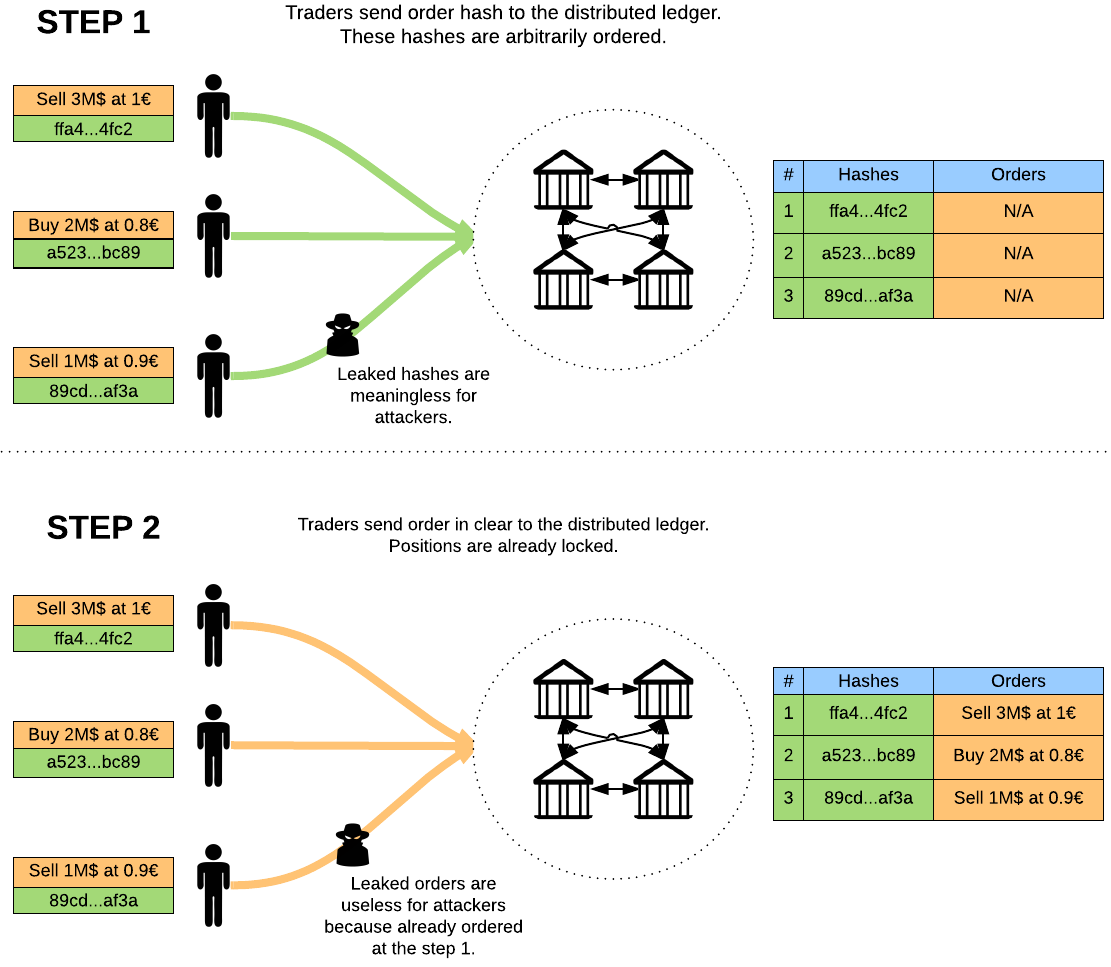
Cryptographic hashes are the mathematical equivalent of fingerprints. Just as a fingerprint is a unique identifier of a person, a hash is a unique identifier of some data, such as a text document, image, or offer to buy a stock. One common use-case of hashing is a situation where someone wants you to be able to verify that your copy of a document is an exact replication of their copy. Once you have been given the hash, the document can be sent over an unreliable or untrusted channel, and the hash allows you to verify that you received the correct and unchanged document. The analogy would be using the fingerprint of a person to identify them; you may know nothing else about the person, and the fingerprint does not let you deduce any information about them, but once the person shows up, you can verify that he or she is the one you expected.

In this paper, we use the hash of a financial order to lock in a time priority for that order. The user will craft the order they want to execute and keep it confidential, but publish the hash. No one receiving the hash will be able to deduce any information about the order, but at a later time when the full order is published, it is trivial for anyone to verify that this is the order used to generate the hash. Any modification to the order after publication of the hash, by the creator or others, will be plainly visible. This completely removes the incentive to re-order transactions - the hashes are devoid of meaningful information, removing the incentive to delay orders, and more importantly, the ability to act on any information in other participants’ orders.

*B. Broadcasting the details of the transaction*

The second stage of the process involves broadcasting the details of the transaction. This occurs once the hash has been processed and assigned an index and timestamp. With time priority established, information leakage is not a problem because all market participants will recognize the relationship between the transaction and its corresponding hash value, thus executing them at its reserved time priority. Figure 2 sets out graphically how this general process will work.

An important feature of this process is that it entails a general delay in the execution of orders. To see why, consider a setting in which participants can set out offers to buy or sell an object. If a market receives a hash, then the sender has to be given time to broadcast their actual offer before any other offers can be executed. This can be accomplished with a timeout; after a hash has been written to the ledger, the sender has a certain time interval to broadcast their offer, or their time priority is forfeited. To allow for normal delays, this timeout would be on the order of seconds in a private distributed ledger (or potentially minutes for a public distributed ledger like Bitcoin).



**Figure 2: Hash Functions and Trading in a Distributed Ledger**

**A Market-based Example**

To clarify both the problem and how our proposed solution would work, we illustrate its operation with a simple example based on trading a financial asset (denoted AXAX). Suppose we consider a trading platform that is running on top of a blockchain or distributed ledger. The underlying ledger is decentralized and the platform utilizes on-ledger, decentralized matching of offers (an example of this is currently found in the Counterparty protocol). In this trading platform, Alice wants to buy 100 AXAX at maximum price of $90. She checks the order book and sees the following orders:

* Sell 90 AXAX at $89
* Sell 10 AXAX at $90
* Sell 100 AXAX at $91

These three orders were created by Bob, a malicious trader who owns a mining pool. He watches the AXAX mempool to intercept matching orders. Alice, of course, doesn’t know that and she puts in a market order for 100 AXAX. When Alice’s order arrives in the mempool, Bob creates an order to cancel the 2 first orders on the book and places them in the Block **before** the Alice order. When the market executes Alice’s order, only the last of Bob’s orders is still open so Alice executes at a price of $91, and not at the blended price of $89.10 that she expected, giving her a worse execution of $1900.

With the proposed solution, Alice would first craft a valid offer (she wants to buy 100 AXAX at 90), run a cryptographic hash algorithm, and publish this encrypted offer to the ledger. Alongside the hidden information in the hash, some visible information (termed “in clear”) such as the market place name and the asset name will be published. This information will be used to determine on what queue the offer should be added. Once the hash has been retained by the ledger, a place in the queue is definitely attributed to the Alice offer. Now Alice has the guarantee that Bob will not be able to insert a cancellation order in the queue before her offer. She then publishes to the ledger the offer in clear (i.e. not hashed and so with details visible to everyone). Once the offer details are retained by the ledger, the matching engine will execute the order at the blended price of $89.10 as expected by Alice. The transaction is posted to the ledger based on the time priority established by the publication of the hash.

Two features of this process are worth noting. First, with this protocol, no information leakage occurs because the initial hash revealed no relevant information that could be exploited by Bob before the offer’s place in the queue is guaranteed. Second, once the initial hash is broadcast the market must delay for a short period while the actual order is sent to the market. If Alice does not publish the order in clear before the end of this short interval, then the hash is removed from the queue and she loses any fee she paid to publish the hash.

**Conclusions**

Distributed ledger technology is beginning to revolutionize back-office clearing and settlement of financial assets, but it also has the potential to revolutionize the actual trading of financial assets. Such a development would require a completely new form of market, one where order execution is handled not by a centralized entity but in a decentralized manner not controlled by any one organization. Before such a market structure can evolve, however, it will have to resolve some basic problems inherent in decentralized trading. As we discussed in this paper, one such problem is its susceptibility to information leakage and front-running. This problem arises from the lack of time priority in the process of writing transactions to the distributed ledger, opening the door to the manipulation of trade order and the insertion of trades based on information in others’ orders.

Our paper shows one way to solve the problem of information leakage by using a cryptographic hash function to reserve a transaction’s time priority, and once this is established following with a second message giving the order’s details. In effect, we use an order’s “fingerprint” to hide its “footprint” in the distributed ledger. We believe this is an innovative approach to dealing with information leakage in a distributed ledger setting. Our hope is that this approach can also set the stage for trading in distributed ledger themselves, opening the way to even greater applications of the distributed ledger technology.

**References**

“A Next Generation Smart Contract and Decentralized Application Program”, available at <https://github/ethereum.com/wiki/White-Paper>.

“Counterparty introduces truly trustless games on the blockchain”, July 2014, available at <http://counterparty.io/news/introducing-fully-trustless-games-on-block>

Demirgas, M. and K. Kulkarni, “Beyond True Time: Using Augmented Time for Improving Spanner, available at [www.cse.buffalo.edu/~demirbas/publications/augmentedTime.pdf](http://www.cse.buffalo.edu/~demirbas/publications/augmentedTime.pdf)

Egelund-Muller, B., Elsman, M., Henglein, F., and O. Ross, 2016, “Automated Execution of Financial Contracts on Blockchains”, Working Paper, University of Copenhagen.

FINRA, Distributed Ledger Technology: Implications of Blockchain for the Securities Industry, 2017, available at http://www.finra.org/sites/defualt/files/FINRA\_Blockchain\_Report.pdf

Harvey, C., “Cryptofinance”, 2016, available at SSRN: https://ssrn.com/abstract=2438299

Nakomoto, S., A Peer-to-Peer Electronic Cash System, White Paper, 2008, available at <https://bitcoin.org/bitcoin>

Pinea, A. and W. Ruttemberg, “Distributed ledger technologies in securities post-trading: Revolution or evolution?” April 2016, Occasional Paper Series of the European Central Bank, available at https://www.ecb.europa.eu/pub/pdf/scpops/ecbopl172.en.pdf

UK Government Office for Science, 2016, Distributed Ledger Technology: Beyond the Block Chain, available at https://www.gov.uk/government/...data/.../gs-16-1-distributed-ledger-technology.pdf

**Appendix - Hash Commitments in Decentralized Trading**

## Decentralized trading

This is trading, including offer matching, running on top of a decentralized system, like Bitcoin’s blockchain or a permissioned, private ledger like Symbiont Assembly. Every offer is published on the underlying ledger in the plain, but due to the decentralized nature of the system, the offers are not immediately assigned a time priority. As such, they are not guaranteed to be processed by the trading system in any particular order.

This allows third parties to publish conflicting offers after becoming aware of the content of an offer, and through random luck or collusion have their offer processed before the one with which it conflicts. In other words, decentralized trading platforms are inherently exposed to front running at a much larger degree than traditional centralized markets.

## Hash Commitments

A hash is a code generated from and tied to any arbitrary set of data. It is considered next to impossible to guess without having access to the corresponding data, and it is similarly hard to reverse the process to uncover the corresponding data. It is also statistically impossible for two different sets of data to generate the same hash, even if the two sets of data are very similar. Most current computer security relies on these assumptions holding, even when actively challenged by intelligent attackers with large resources.

With a naive approach to hashing, data that is essentially identical could produce the same hash, potentially leaking some information about the trader’s intentions if that data is an offer to trade on a market. For instance, it can be a repeated order, either at a later time or to a different market, be recognized, and have its content leaked. This is a well-known feature of hashes, and it can be worked around trivially. The data input into the hashing function is extended with a field without any semantic meaning, typically a timestamp or a random number. This field is stripped before parsing the data, but has the effect that otherwise identical data now produce different hashes if different values are put into this extra field.

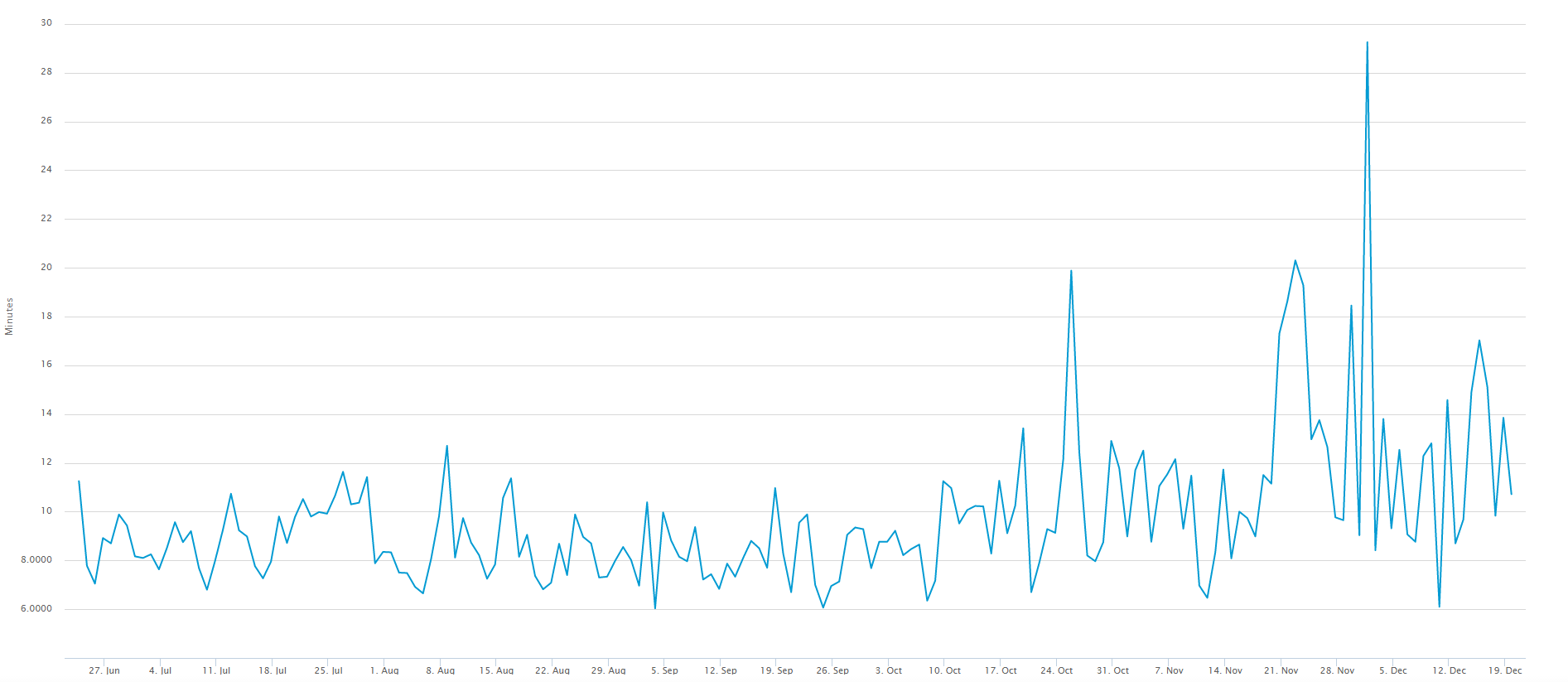
A hash commitment is the use of a hash to commit to a set of data without revealing the data. Simply put, the actor who knows the data generates a hash from it and publishes the hash to whomever is interested in the commitment. At a later stage, when the data becomes known to other parties, they can verify the relationship between the data and the hash, and thus verify that the data has not been changed since the commitment.

## Two-stage offer publication

We propose relying on hash commitments to hide information in decentralized offer matching. A trader wishing to publish an offer to the decentralized market, potentially intending to match against an offer already on the order book of the market, will craft an offer according to the trading protocol of the decentralized market but, crucially, postpone the publication of the offer.

Instead, the trader will publish a hash of the offer to the underlying ledger, the hash commitment. Once this commitment has been irrevocably written to the ledger (and thus assigned time priority), the full offer is published to the same ledger.

The decentralized market needs to be extended to expect this two-stage offer publication. It will perform the offer matching after a delay, and with time priority of the offers inherited from the matching hash commitments. Commitments published, but missing a matching offer at the end of this delay, are considered abandoned and are discarded without affecting the market. The market may never know what the intentions of these commitments were, but will still charge the originator a trading fee to discourage abuse.



**Figure 1 - Median Confirmation Time**

The Figure gives the median time in minutes that it takes for a transaction to be accepted into a mined block and added to the Bitcoin public distributed ledger. The data are for the period June 27, 2016 – December 19, 2016. The transactions here are only those that include transactions fee. The chart is from https://blockchain.info/charts/median-confirmation-time?timespan=180days.

1. For discussion of the potential uses for distributed ledgers see UK Office for Science, *Distributed Ledger Technology: Beyond the Block Chain* [2016]; Financial Industry Regulatory Authority, *Distributed Ledger Technology: Implications of Blockchain for the Securities Industry*, January 2017; Pinea and Ruttenberg [2016]; and Egelund-Muller, Elsman, Henglein, and Ross [2016]. [↑](#footnote-ref-1)
2. The Counterparty system for decentralized finance on the Bitcoin blockchain has previously implemented a protocol for completely trustless multi-party games, with crypto currency wagering, in the context of the rock, paper, scissors game. See <http://counterparty.io/news/introducing-fully-trustless-games-on-block> . Our approach draws on this work for the broader application to trading in distributed ledgers. [↑](#footnote-ref-2)
3. The notion of trust is a crucial concept here. Participants use centralized systems that they view as trustworthy, or are made trustworthy through the efforts of regulators. Decentralized systems like the Bitcoin blockchain rely on unknown (and, thus, untrusted) parties, and have no regulators or other outside interference to establish trust. What allows such untrusted systems to operate successfully is the cryptographic-based rules defining the blockchain technology. See Nakomoto [2008] for discussion of how such a trustless system can operate. [↑](#footnote-ref-3)
4. See Harvey (2016) for a thoughtful discussion of the mechanics of distributed ledgers. [↑](#footnote-ref-4)
5. This problem is at the heart of “TrueTime” or the solution used both internally and externally by Google in which time is measured by ranges instead of in discrete units. For discussion, see Demirbas and Kulkarni (2013). [↑](#footnote-ref-5)
6. This will generally be true but would not be the case if a cryptographic privacy solution is employed or the transactions are not broadcast globally and there not available for matching, [↑](#footnote-ref-6)
7. To make matters worse, the Bitcoin network will sometimes experience rollbacks of previously written blocks. For large transactions it’s customary to wait for the block containing the transaction to be followed by 2-5 blocks (3-6 *confirmations*), at which point a rollback is considered highly unlikely (the likelihood of a rollback of a given block falls quickly as new blocks are added after it in the chain). [↑](#footnote-ref-7)
8. Note this problem also arises in gaming applications. The Counterparty system protocol noted earlier addresses this problem in the context of the rock-paper-scissors game using roughly the scheme described in this paper. Network participants would broadcast a wager and the hash of a game move (plus a nonce). Then the protocol would match the move with that of another player (with matching wager size and asset). Finally, both participants would reveal their moves and nonces. The winner would receive the sum of the two wagers, minus a transaction fee. If either player did not broadcast his move within a time limit, he would be forced by the decentralized protocol to forfeit his wager. In the case of a tie, the wagers were returned. [↑](#footnote-ref-8)
9. Even this information may be obscured in some settings. If the ledger is used for other things than trades, a hash can belong to a different protocol. Furthermore, without context, a cryptographic hash is indistinguishable from random data. [↑](#footnote-ref-9)
10. There is an extensive literature on cryptographic hashes (see, for example, the National Institute of Standards and Technology’s (NIST) 2015 SHA-3 release [http://nvIpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf](http://nvipubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf)). There is also a large literature on theoretical and practical attempts at attacking the various hashes. Cryptographic hashing algorithms generally have a life cycle where they are released (SHA-3), used (SHA-2), found susceptible to theoretical attacks (SHA-1), and finally, practical attacks are demonstrated (SHA-0, SHA-1, MD5). The brief window of time where the hash in the system proposed in this paper is relevant, and the ease with which the algorithm can be upgraded, puts very lenient requirements on the strength of the hashing algorithm employed. [↑](#footnote-ref-10)