Footprints on the Blockchain:

Information Leakage in Distributed Ledgers

by

Rune Tevasvold Aune, Maureen O'Hara, and Ouziel Slama*

December 2016

This paper examines information leakage 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 cryptology-based approach for solving information leakage problems in distributed ledgers that relies on using a "hash" (or mathematical identifier) to secure time-priority followed by a second communication revealing more features of the underlying market transaction. Solving the information leakage problem greatly expands the potential applications of private distributed ledger technology.

* Rune Aune is Senior Software Engineer, Symbiont IO; Maureen O'Hara is the Robert W. Purcell Professor of Finance at Cornell University, Professor at UTS (Australia), and an Advisor to Symbiont IO; Ouziel Slama is Principal Software Engineer, Symbiont IO. The authors would like to thank Adam Krellenstein for helpful discussions.

Footprints on the Blockchain:
Information Leakage in Distributed Ledgers

1. Introduction

Advocates of the blockchain, the distributed ledger technology that underlies Bitcoin, see an almost limitless future for the innovation. Apart from its role in cryptocurrencies, distributed ledger technologies are being developed to handle stock trades, clear and settle leveraged loans, handle catastrophe reinsurance, keep track of land titles and transactions, track 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 a decentralized distributed ledger technology in which an immutable single record emerges of the various transactions encased in the blocks. The ability to trade assets in such a setting, along with the emergence 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]

While not disputing this rosy outlook, we argue in this paper that this new technology comes with some very old problems. In particular, a fundamental problem facing traders in traditional market settings 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

[1] An excellent discussion of the potential uses for distributed ledgers is UK Office for Science, Distributed Ledger Technology: Beyond the Block Chain [2016].

algorithmic and technological solutions are dedicated to 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 blocks in the 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 participants in a distributed ledger to potential front-running and manipulation.

A second, and perhaps more important, contribution of this paper is to suggest a solution. We propose a cryptology-based approach for solving information leakage problems in distributed ledgers. Our two-part process uses a "hash" (or mathematical identifier) 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 curtailed. This solution essentially 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 are focused mainly on 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 new cryptography-based tools can provide a solution to an old problem in a new setting.

2. Information Leakage in Distributed Ledgers

To understand why information leakage can occur, it is useful to first set out the distinction between centralized and decentralized systems, and how their differences affect transactions processing. Centralized systems (for example, exchanges) are straightforward: they have a single 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 single party, and instead it relies on a network of independent parties geographically dispersed executing an algorithm to process transactions. 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, it is the limitations of this sequencing process that sets the stage for information leakage to occur.

A. Transaction processing in decentralized systems

Due to transmission delays in a network, it is impossible to reach a global agreement on the actual order that transactions happen in a decentralized system. Such latency issues are an all too familiar problem in the current high frequency market structure characterizing equity 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 equity trading 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 can be distributed globally, making transmission delays an inevitable feature of the system.[2]

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.[3] In the period between the broadcast of a Bitcoin transaction and its being processed by the network, for example, there are typically 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 will be observable by all or parts of the network in this potentially lengthy period. 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 has been 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

[2] See Harvey (2016) for a thoughtful discussion of the mechanics of distributed ledegers.
[3] 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).

to offer the guarantee that no new transactions may 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, but its severity depends upon the validation process employed. In particular, public distributed ledgers and private distributed ledgers use different validation methods and, as we discuss below, these methods can affect the information leakage problem.

B. Transaction processing in Bitcoin and Ethereum

In a public distributed ledger system like the Bitcoin and Ethereum networks, transactions are processed by "miners" (essentially high powered computers) who write these transactions into blocks, and collect the fees associated with each transaction. There are numerous miners on the network, and which one writes a specific block is random, with the probability of doing so equal to their share of mining power relative to the total. When a transaction is broadcasted, 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.[4] As Figure 1 shows, these delays in Bitcoin have typically been around 10 minutes, but

[4] 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).

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 will seek to optimize their own returns by including as many transactions as they can in each block (thus collecting the fees). This category of miner will not have any incentive to manipulate the relative order of transactions, but neither will they have any interest of keeping them in any particular order. Thus, transactions containing conflicting orders may be shuffled around arbitrarily for a time period of multiple minutes.

Other miners, however, may have motivations 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.

C. Transaction processing in leader driven decentralized systems

In contrast to the public distributed ledgers described above, there can also be private distributed ledger systems. In such systems (examples include Symbiont Assembly or Axoni), an elected leader is responsible for processing transactions. These leader-driven systems don't necessarily operate on blocks, but the transactions are written to a distributed ledger. Relative to Bitcoin, the delay between a transaction broadcast and it being written to the ledger is much

shorter, often on the order of seconds for a global system. That's still more than enough time for other participants on the network to broadcast their own transactions with conflicting orders in them.

Like Bitcoin miners, the 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 difficult to prove suspected willful reordering.

D. Distributed systems and time

The issues outlined above are due to two features of these systems: 1) they are decentralized, meaning no central authority can force miners or leaders respectively to process transactions in a particular order, and 2) they are distributed, meaning that the nodes making up the system are distributed geographically, and due to network delays 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 systems will be unable to preserve correct ordering globally for transactions happening near each other in time. For example, Google's "TrueTime" system can guarantee relative ordering of events that happen at least 7 milliseconds (ms) apart globally but for events happening closer in time, the

actual ordering cannot be guaranteed. This lack of time priority sets the stage for information leakage and potential front-running in distributed ledgers.

3. A Proposed Solution

The basic problem to be solved is straightforward: remove the ability of miners or leaders to profit from trading on the information in your transaction. Once your transaction has been written to the distributed ledger, the problem is largely ameliorated, but it is in the preceding processing stage that information leakage can occur. So a natural starting point for a solution is to consider changes to how this initial processing takes place, with the aim of establishing strict time priority for transactions.

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.[5] Broadcasting that hash value and waiting for it to be processed by the system will reserve an order's time priority. Once the priority has been secured, the order itself is broadcast. At this stage, the information leakage is not of any concern as all involved markets 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 more explanation given in the Appendix.

[5] 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.

A. Cryptographic hashes - the 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.[6] 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 this 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 doesn't 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.

[6] 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://nvlpubs.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, 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 put very lenient requirements on the strength of the hashing algorithm employed.

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, keep it confidential but publish the hash. No one receiving the hash will be able to deduce any information about the order, but when, at a later time, 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 removes the incentive to re-order transactions completely - 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 sending in 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 is most easily handled 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).

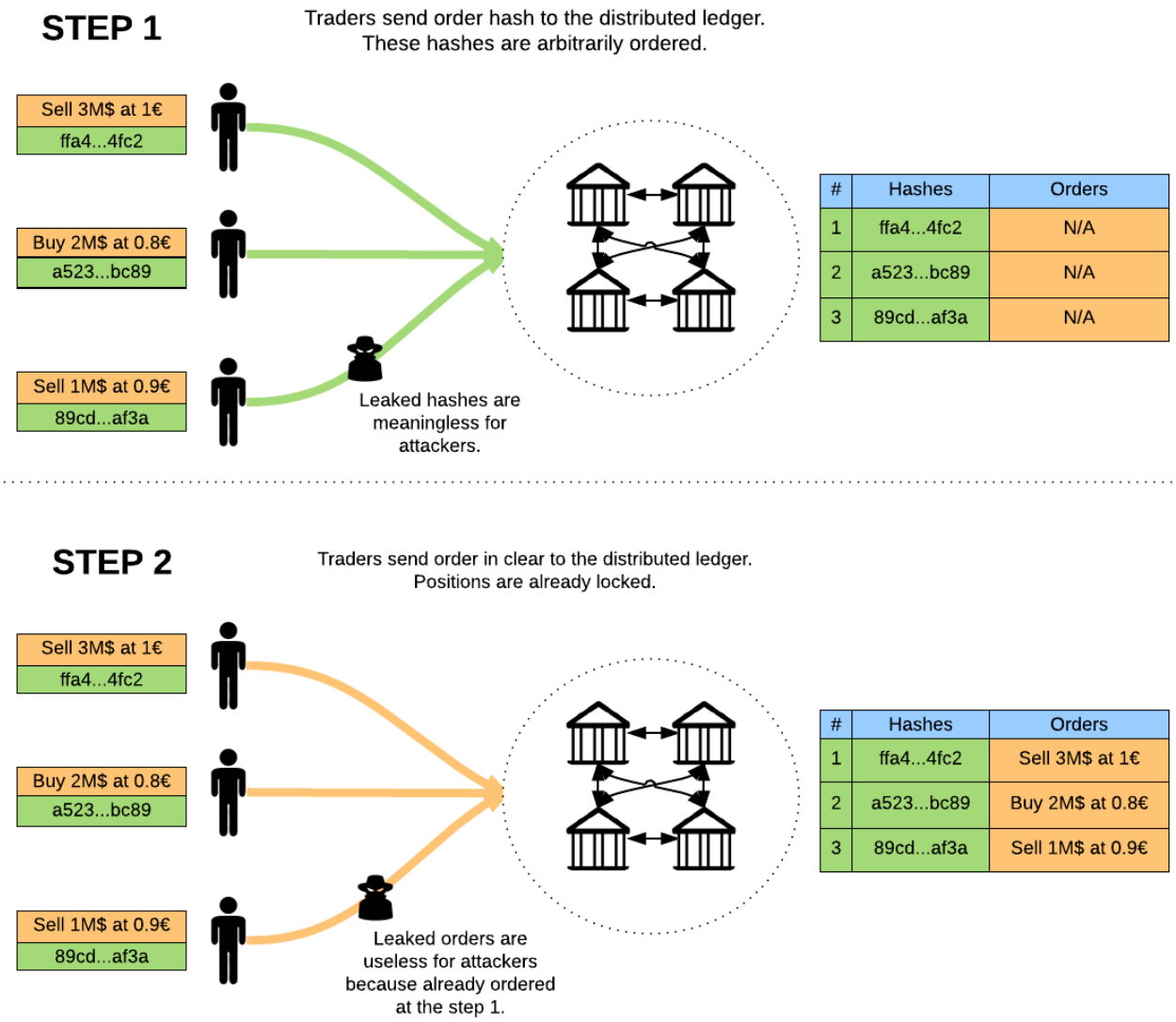

Figure 2: Hash Functions and Trading in a Distributed Ledger

4. 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 Bob order 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 (termed "in clear") information 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 (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 89.1 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 has been 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, the hash will be removed from the queue and she will lose any fee paid to publish the hash.

5. Conclusions

The distributed ledger may be the technology that brings us a completely new form of market, one where order matching itself 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 have 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.

We have shown one way to solve the problem of information leakage by using a cryptographic hash function to reserve a transaction's time priority, and then once this is

established following with a second message giving the order's details. In effect, we are using 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 paper shows how a cryptology-based trading mechanism may play a role in facilitating trading in distributed ledger markets.

References

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

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

Harvey, C., Cryptofinance, January 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

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 to 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 it conflicts with. 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 extremely unlikely 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 parameters holding, even when actively challenged by intelligent attackers with large resources.

With a naive approach to hashing, data that is essentially equal 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 publish the hash to whoever 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 who wishes 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 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.

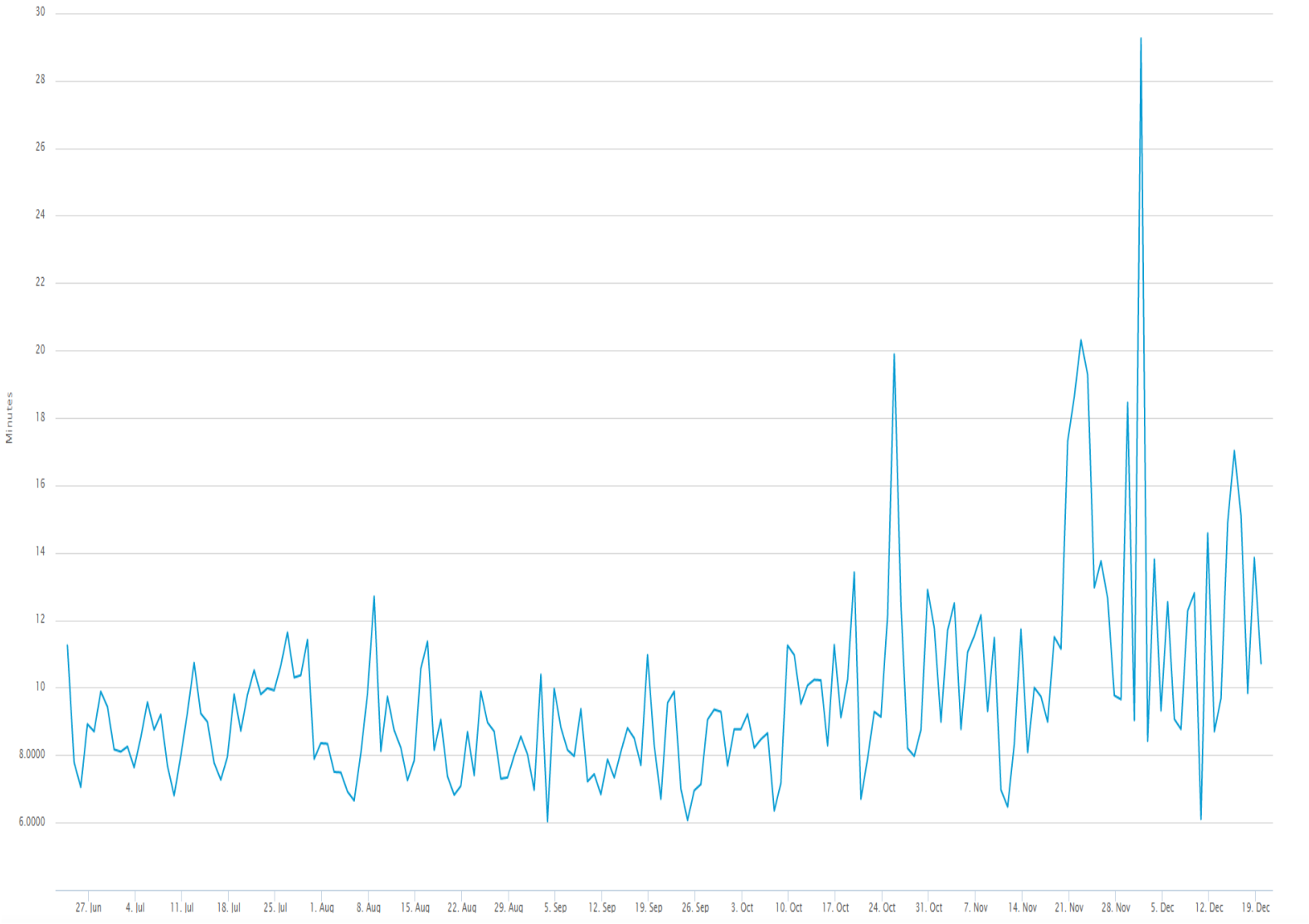

Figure 1 - Median Confirmation Time

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.