Footprints on a Blockchain: *Trading and Information Leakage in Distributed Ledgers*

Rune Tevasvold Aune, Adam Krellenstein, Maureen O'Hara and Ouziel Slama

JOT 2017, 12 (3) 5-13
doi: https://doi.org/10.3905/jot.2017.12.3.005
http://jot.iijournals.com/content/12/3/5

Institutional Investor Journals
1120 Avenue of the Americas, 6th floor,
New York, NY 10036, Phone: +1 212-224-3589

Footprints on a Blockchain: *Trading and Information Leakage in Distributed Ledgers*

Rune Tevasvold Aune, Adam Krellenstein, Maureen O'Hara, and Ouziel Slama

Rune Tevasvold Aune
is a senior software engineer with Symbiont.io, Inc. in Trondheim, Norway.
rune.aune@symbiont.io

Adam Krellenstein
is the chief technology officer with Symbiont.io, Inc. in New York, NY.
adam@krellenstein.com

Maureen O'Hara
is the Robert W. Purcell Professor of Finance in the Johnson Graduate School of Management at Cornell University in Ithaca, NY.
mo19@cornell.edu

Ouziel Slama
is a principal software engineer with Symbiont.io, Inc. in Bordeaux, France.
ouziel.slama@symbiont.io

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]

Although we do not dispute this optimistic outlook, we argue in this article that this new technology comes with some very old problems. In particular, in traditional market settings, traders face the fundamental problem of hiding 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 timestamped, there is no actual time priority in the process of getting a transaction validated and onto the distributed ledger. One contribution of our article is showing 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.

The second, and more important, contribution of this article is a suggested solution to the problem. We propose a cryptographic approach to solving information leakage problems in trading on distributed ledgers.[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 the 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 some distributed ledger applications. To date, proposed uses of distributed ledgers focus mainly on posttrade clearing and settlement, in which 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 (e.g., exchanges) are straightforward: They have a single trusted party, usually in a single location, that processes transactions and assigns timestamps to them. Decentralized systems (e.g., the Bitcoin blockchain) are much more complex: There is no trusted party; it instead relies on a network of untrusted, independent parties executing an algorithm to process transactions.[3] The most relevant part of this process for the focus of our article is the sequencing of transactions, in which 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.

Transaction Processing in Decentralized Systems

Because of 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 that characterizes equity or futures trading. The problem in a decentralized system is more severe, however, because different nodes in the network see transactions appear at different times, and 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.[4] 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 regard to the transaction's order relative to other transactions in the same state or what timestamp the transaction will eventually be assigned.

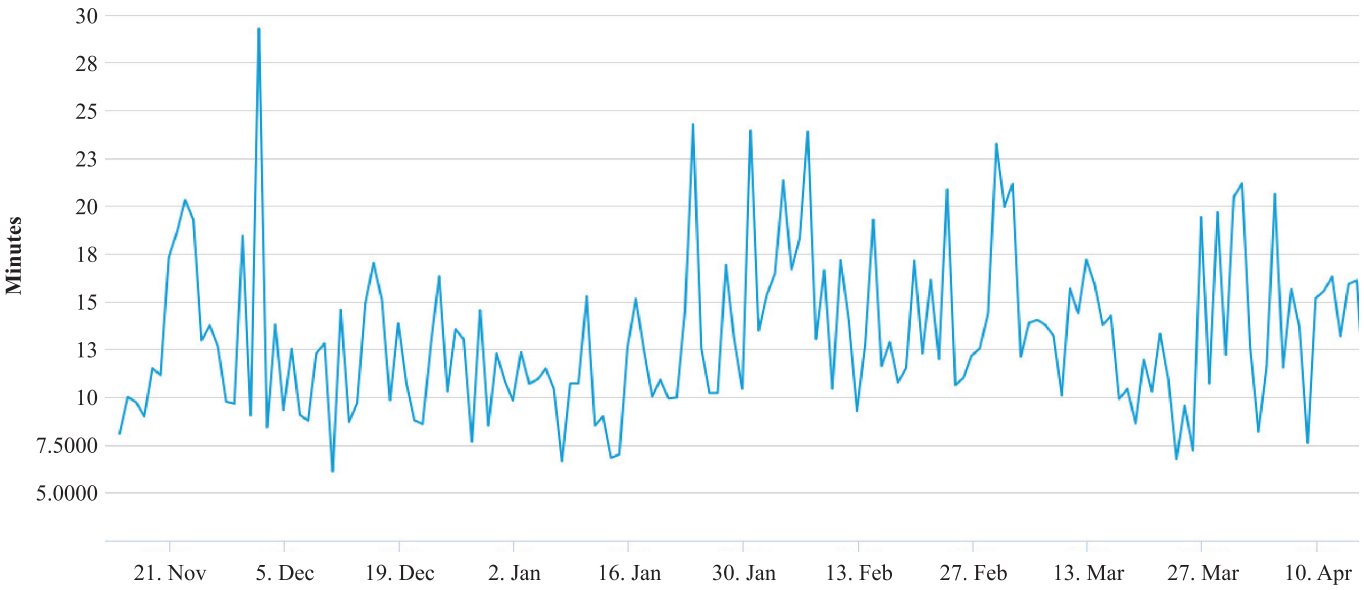
Crucially, the content of any transaction (where we are using *transaction* in a broader sense to include events such as a trade, a quote, or indication of interest) that will be posted to the distributed ledger is observable by all or part of the network in this period.[5] This visibility allows other participants to act on 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, who are untrusted, may also be able to alter the relative ordering of transactions arbitrarily and potentially to their own benefit. Furthermore, depending on the extent of these alterations, 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 will discuss, public distributed ledgers use miners and private distributed ledgers use an elected leader to validate transactions, but the basic problem is the same regardless of the method used: One party is temporarily trusted with ordering the transactions because there is no global agreement on a natural ordering.

Transaction Processing in Public Ledgers

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

EXHIBIT 1
Median Confirmation Time



Notes: The exhibit 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 November 15, 2016 to April 15, 2017. The transactions here are only those that include transactions fees.
Source: https://blockchain.info/charts/median-confirmation-time?timespan=180days.

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 their doing so being 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 it sees 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.[6] As Exhibit 1 shows, these delays in Bitcoin have typically been around 10 minutes, but they can be much longer because there is 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 it 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 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. The anonymity of most miners 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 Private Ledgers

In contrast to the public distributed ledgers just described, there are also private distributed ledger

systems. In such systems (e.g., 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 containing conflicting orders.

Like Bitcoin miners, the elected leader has the freedom to order transactions relative to each other, temporarily taking responsibility for providing a canonical ordering of transactions when there is no trusted party to do so. 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, the 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.

Distributed Systems and Time

The issues outlined are due to two features of distributed ledger systems: (1) They are distributed, so as a result of network delays, the geographically disbursed nodes may receive transactions in a different order; and (2) they are decentralized, so there is no central authority that can be trusted to resolve these differences. This lack of global time priority sets the stage for information leakage and potential front-running in any distributed ledger.

A PROPOSED SOLUTION

The basic problem is straightforward: How can we remove the ability of miners or leaders to profit from trading on the information in your transaction? The problem is largely ameliorated once a transaction is written to the distributed ledger, 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. Therefore, 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.[7]

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 later), producing an identifier that is mathematically linked to the order but not exposing any details of the order itself, other than the fact that some trade is desired.[8] 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 because 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.

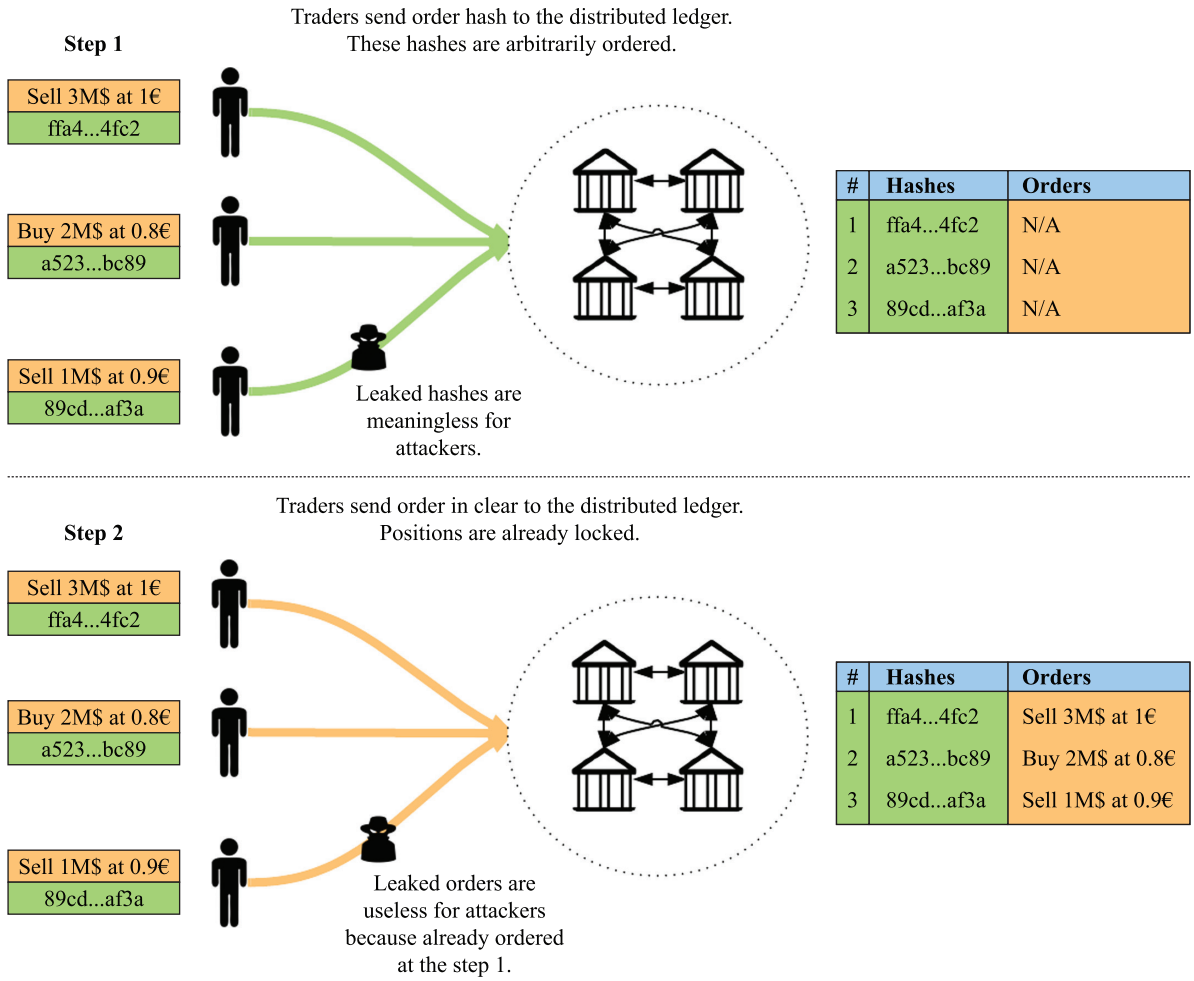
Cryptographic Hashes—A Transaction's Fingerprint

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.[9] 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 in which someone wants a recipient to be able to verify that a copy of a document is an exact replication of his or her copy. Once given the hash, the document can be sent over an unreliable or untrusted channel, and the hash allows the recipient to verify that he or she received the correct and unchanged document. The analogy would be using a person's fingerprint to identify him

EXHIBIT 2

Hash Functions and Trading in a Distributed Ledger



or her; nothing else may be known about the person, and the fingerprint does not allow any information to be deduced, but once the person shows up, he or she can be verified as the person expected.

In this article, we use the hash of a financial order to lock in a time priority for that order. The user will craft the order he or she wants 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 reorder transactions—the hashes are devoid of meaningful information, thus removing the incentive to delay orders and, more importantly, the ability to act on any information in other participants' orders.

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 their reserved time priority. Exhibit 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 the 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 the offer, or the time priority is forfeited. The network as a whole verifies that the offer matches the hash broadcast previously. To allow for normal delays, this timeout would be on the order of seconds in a private distributed ledger or minutes for a public distributed ledger like Bitcoin.

In a similar vein, cancellations would have to be subject to the same restrictions and delay as entering an order. If you have an active order on the book (that you would like to cancel), someone might publish a hash of an offer that would match your order. Your cancellation then would have to be delayed long enough that all outstanding hashes that may be linked to a matching offer have time to process. For complete symmetry, the cancellation could also have to be published as a hash first.

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 uses 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 a 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, does not 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 two first orders on the book and places them in the block before Alice's 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 $1,900.

With the proposed solution, Alice would first craft a valid offer (she wants to buy 100 AXAX at 90), then 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 the queue to which the offer should be added. Once the hash has been retained by the ledger, a place in the queue is definitely attributed to Alice's 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 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 Alice expected. The transaction is posted to the ledger based on the time priority established by the publication of the hash.

Three 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. Latency would be on the order of the confirmation time of the ledger in question.[10] The only requirement is that once the hash has been broadcast and included in the global transaction log, its position therein must be indisputable. Third, there must be a strong, out-of-band penalty for Alice if she fails to publish the order in the clear before an agreed-upon timeout. Without this penalty, the system could easily be abused by publishing hashes for multiple orders and only revealing the plaintext of the hashes of offers that followed profitable market movements within the timeout interval. The disincentives must suffice to make it impossible to make money this way. The faster the network and the shorter the timeout, the smaller the potential profit, suggesting that market design features may alleviate this problem.

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 in which 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 article, one such problem is the 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 article shows one way to solve the problem of information leakage by using a cryptographic hash function to reserve a transaction's time priority, following with a second message giving the order's details once priority is established. 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 ledgers themselves, opening the way to even greater applications of the distributed ledger technology.

APPENDIX

HASH COMMITMENTS IN DECENTRALIZED TRADING

Decentralized Trading

Decentralized trading 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 because of 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 lack of priority 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 to 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 nearly 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 true, even when actively challenged by intelligent attackers with large resources.

With a naive approach to hashing, data that are essentially identical could produce the same hash, potentially leaking some information about the trader's intentions if those data are 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 data are parsed but has the effect of making otherwise identical data 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 become known to other parties, these parties can verify the relationship between the data and the hash and thus verify that the data have not been changed since the commitment.

A hash is verified by simply recomputing it from plaintext value once that has been posted and comparing it to the previous value. Verification would be done by all of the network participants that are involved in achieving distributed consensus. Hashes are extremely fast to calculate, so flooding the system with hashes is no greater danger than flooding the system with any other kind of data.

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. The market will perform the offer matching after a delay and with time priority of the offers inherited from the matching hash commitments. To avoid potential manipulation, traders would have to be sufficiently disincentivized through fees and other inducements from publishing a hash for which they do not eventually reveal the plaintext value.

ENDNOTES

The authors thank Caitlin Long for helpful comments.

[1]For a discussion of the potential uses for distributed ledgers, see U.K. Office for Science [2016]; Financial Industry Regulatory Authority [2017]; Pinna and Ruttenberg [2016]; Harvey [2016]; and Egelund-Muller et al. [2016].

[2]The Counterparty [2014] system for decentralized finance on the Bitcoin blockchain previously implemented a protocol for completely trustless multiparty games, with cryptocurrency wagering, in the context of the rock, paper, scissors game. Our approach draws on this work for broader application to trading in distributed ledgers.

[3]The notion of trust is a crucial concept here. Participants use centralized systems that they view as being trustworthy or being 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 a discussion of how such a trustless system can operate.

[4]This problem is at the heart of *TrueTime*, or the solution used both internally and externally by Google, in which time is measured in ranges instead of in discrete units. For discussion, see Demirbas and Kulkarni [2013].

[5]This will generally be true but would not be the case if a cryptographic privacy solution were employed or if the transactions were not broadcast globally and not available for matching.

[6]To make matters worse, the Bitcoin network will sometimes experience rollbacks of previously written blocks. For large transactions, it is customary to wait for the block containing the transaction to be followed by two to five blocks (three to six *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).

[7]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 article. Network participants would broadcast a wager and the hash of a game move (plus a nonce). The protocol then would match the move with that of another player (with a 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 or her move within a time limit, he or she would be forced by the decentralized protocol to forfeit the wager. In the case of a tie, the wagers were returned.

[8]Even this information may be obscured in some settings. If the ledger is used for things other than trades, a hash can belong to a different protocol. Furthermore, without context, a cryptographic hash is indistinguishable from random data.

[9]There is extensive literature on cryptographic hashes (e.g., 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 body of literature on theoretical and practical attempts at attacking the various hashes. Cryptographic hashing algorithms generally have a life cycle in which 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 in which the hash in the system proposed in this article is relevant, and the ease with which the algorithm can be upgraded, puts very lenient requirements on the strength of the hashing algorithm employed.

[10]The latency of a permissioned distributed ledger is on the order of one second if the network is widely geographically distributed and less if geographically concentrated. Whether a blockchain has a latency advantage depends on the particular market in question: There are existing financial markets with latency both much greater than and much less than this, and blockchain technology can theoretically be used to handle the trading of all of them.

REFERENCES

Counterparty. "Counterparty Introduces Truly Trustless Games on the Blockchain." July 2014. http://counterparty.io/news/introducing-fully-trustless-games-on-block.

Demirbas, M., and K. Kulkarni. "Beyond True Time: Using Augmented Time for Improving Spanner." 2013. www.cse.buffalo.edu/~demirbas/publications/augmentedTime.pdf.

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

Financial Industry Regulatory Authority. "Distributed Ledger Technology: Implications of Blockchain for the Securities Industry." 2017. http://www.finra.org/sites/defualt/files/FINRA_Blockchain_Report.pdf.

Harvey, C. "Cryptofinance." SSRN, 2016. https://ssrn.com/abstract=2438299.

Nakomoto, S. "A Peer-to-Peer Electronic Cash System." White paper, 2008. https://bitcoin.org/bitcoin.

Pinna, A., and W. Ruttenberg. "Distributed Ledger Technologies in Securities Post-Trading: Revolution or Evolution?" Occasional Paper Series of the European Central Bank, April 2016. https://www.ecb.europa.eu/pub/pdf/scpops/ecbop172.en.pdf.

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