

The paper on Blockchain Frontrunning

No Author Given

Concordia University

Abstract.

1 Introduction

2 Preliminaries

2.1 What is Front-running?

Front -unning is analogous to any course of action during which a person benefits from prior access to inside or confidential market information about upcoming transactions and trades. This problem can occur in both financial and non-financial systems, however, it is more noticeable within the trading and exchange systems as it was originated back in stock markets. In the old times, all the trades were executed on papers, so in case of receiving a large order from a client, the broker might say this loud and other people around the table could be informed. Therefore, a malicious trader would now run in front of that order and put his own transaction in between (before the trade was executed) and profit from the price increase of the stock. In other words, a group of market participants obtain non-public market information which allows them to front-run other users trades by putting their orders a head of those trades and benefit from advance knowledge of pending client orders. The two significant factors which cause the front-running practice to happen within the financial markets are (i) imperfect competition and (ii) liquidity uncertainty. **should I explain more?!** Any sort of front-running activity within the financial markets is considered as an unethical and illegal practice as it is unfairly beneficial for a few market participants who have the privilege of acting on this information and taking advantages at the expense of the investor.

2.2 How are Blockchain Applications Vulnerable to front-running?

Blockchain technology has received an exceptional amount of attention since it was first introduced as the underlying technology of the cryptocurrency Bitcoin in Satoshi Nakamoto's (pseudonymous) 2008 white paper [?]. Many decentralized applications that are nowadays built on top of this technology represent the significance of the blockchain as it completely eliminates the single points of failures within any systems. However, blockchains have some inherent characteristic that leave them vulnerable to *front running* behaviour. Although this data

structure is known to be publicly visible to every network participants, information is layered and some of them are accessed only by insiders. Any network participants that runs a full node on the blockchain is able to obtain those information and as mentioned in Section ??, this leaves the application vulnerable to front running. Decentralized exchanges are a group of blockchain financial applications where front running can be executed. Bancor [?] is an example of such systems in which front-runners benefit from potential price increase of the market stocks (details are provided in Section ...). Decentralized namespaces, as blockchain non-financial applications, are another example. Ghazal [?] is such system in which upon seeing a transaction from Alice to register `alice.com`, front runner can go ahead of her and register this domain name. He can later sell `alice.com` to Alice for a higher price and benefit from the price increase. Another application that is vulnerable to front running is Initial Coin Offering, ICO, smart contracts. ICO is a method to distribute tokens based on the deposits of the native coin, Ether in the case of Ethereum. After a few hacking incidents on high valued smart contracts [?], ICOs started to implement restrictions and capped how much funds can be gathered. This scarcity of the initial coins made for a competition to incentivize big investors to get in and buy the tokens at a discounted price and sell them to late comers on the open markets [?, ?]. ICOs started to experiment with different fair capping methods, such as reverse dutch auction, dynamic ceilings, *e.g.*, [?]. This competition made ICOs a good opportunity for front-running attacks.

3 Related Work

As the traditional front-running was originated to trading financial instruments most of the literature are focused on financial aspects of the markets.

3.1 Historical Context / Classic front-running

front-running was first appeared on the Chicago Board Options Exchange (CBoE) [?], and was identified by *Securities Exchange Commission* in 1977 as the following:

The practice of effecting an options transaction based upon non-public information regarding an impending block transaction¹ in the underlying stock, in order to obtain a profit when the options market adjusts to the price at which the block trades. [?]

On the years after there were ongoing discussions between self-regulatory exchanges (*e.g.*, CBoE) and SEC to regulate, detect and define laws and regulations to deal with front-running [?], with SEC stating:

¹ Block in the stock market is large number of shares, 10 000 or more, to sell which will heavily changes the price

It seems evident that such behaviour on the part of persons with knowledge of imminent transactions which will likely affect the price of the derivative security constitutes an unfair use of such knowledge.²

As defining what exactly is considered illegal front-running required more knowledge of how these new markets behave, *CBoE* and other exchanges (and brokers) issued educational circulars for their members asserting that front-running violates existing rules, with some examples of what is considered front-running. However difference of opinion regarding the unfairness of front running activities, insufficient exchange rules and lack of a precise definition in this area resulted in no action by self-regulator organizations *sec1978optionsmarket*.

Further reading on the early details on the history and challenges of detecting and regulating front-running can be found in [?].

Initially the front-running policies only applied to certain option markets, later on in 2002, the rule was refined to include the same prohibitions to security futures [?], which then in 2012 with the new amendment, FINRA Rule 5270, the front-running rule was extended to cover trading in an option, derivative, or other financial instruments overlying a security with some exceptions [?, ?].

3.2 Recent Literature on Blockchain front-running

There are only a few research carried out on the front-running subject in the context of blockchain and decentral ledgers. Aune *et al.* discuss how the lack of time priority between broadcasting a transaction and its validation by miners on a blockchain based system would lead to the market information leakage [?]. As mentioned in Section ??, the information leakage results the entire system to be vulnerable to the front-running. They also propose a cryptographic approach to prevent any decentralized system from front-running. In this approach, which is similar to commit and reveal, a transaction hash (fingerprint) is broadcast to the network to secure time priority while not revealing the information. Once the hash is processed and assigned an index and timestamp, the transaction details are broadcast. This makes the transaction and its hash to be executed at their reserved time priority and not vulnerable to front-running.

Malinova and Park discuss different design settings for financial markets and trading platforms on decentral ledgers [?]. They argue that blockchain brings a new options of transparency which affects traders behaviour, specially in the presence of intermediary front-runners. They model indirect trading costs and use the economic literature on search and trading at decentralized exchanges in their design settings to make front-running costly and inefficient.

Breidenbach *et al.* [?] face the front-running issue in the context of automating bug bounties for smart contracts. Upon submitting the bug bounty to the Hydra smart contract, one can front-run the bug report and claim the bounty instead of the original reporter. They propose *Submarine Commitments* which is

² Securities Exchange Act Release No. 14156, November 19, 1977, (Letter from George A. Fitzsimmons, Secretary, Securities and Exchange Commission to Joseph W. Sullivan, President CBoE).

to conceal the bounty claim and the participatory factor. They claim submarine commitments “are a stronger form of a commit-reveal scheme”.

Some researchers have discussed the double spending attack on the Bitcoin blockchain. In this attack, as the name implies, the adversary spends a coin more than one time. To do so, he would use the coin as an input of a transaction and broadcast it to the network, before the transaction gets validated and executed, he broadcasts another transaction associated with a higher transaction fee which spends the same input $[?, ?]$. This adversarial behaviour is similar to the front-running attack where the attacker would front-run other users by paying a higher gas price when broadcasting his transactions to the network. As mentioned above, this leads to the faster execution of the transaction on the Ethereum blockchain.

4 (On) Blockchain Front-running

4.1 Who Can Front-run?

In general, all the network participants have the ability to front run specific transactions that are sent to the network. However, miners can include any transactions they like into the block they attempt to mine. Thus, they possess special power in terms of front running as opposed to other users of the network. In the following, we discuss and compare the two groups of blockchain potential front-runners.

Miners As mentioned above, since the Blockchain miners are the only parties who can decide on the order of transactions within a block they mine, they can easily intercept and reorder the pending transactions sitting in the mempool, this is known as **transaction reordering** attack. Miners could use this power to attack any smart contract with transaction-ordering dependence vulnerability [?]. For example in an Ethereum based application, a miner learns about the pending *buy* transaction of 1000 units of a token, i.e. TKN, presumably if this transaction goes through, it causes the price the purchased token to increase. So the dishonest miner can step in front of this transaction and place his own buy order ahead of it. He would simply create his *buy* of 1000 TKN and include it within a block and now he mines the previous *buy* transaction of 1000 TKN. Doing so, he would receive a better rate than other network users, and can sell the assets he has received and gain a price advantage at the expense of others. Similarly, a dishonest miner can sell his tokens in front, if he sees a pending *sell* transaction. In an open source blockchain application, there is no rules on how transactions must be ordered and miners are free to send transactions in the order they prefer. However, miners can only front run other transactions (by reordering them) within the block they happen to mine. They could do this also by not broadcasting their own transactions to the network, this makes the miners to be less noticeable to the network when front running. Another example here is in the case of decentral exchanges and order cancellation. Every exchange

should have the functionality to cancel the orders, especially for a volatile market. In this case, when user decides that her order is not profitable anymore, she would send a cancelation transaction. Here the miner sees the cancelation transaction and puts his own order before the cancellation transaction –transaction reordering– to fill the order, potentially profiting from the order. They can also include the canceled transaction in their block to collect the full gas limit used by that transaction [?].

Blockchain users/nodes Any regular (non-miner or miner) user can also front-run other transactions in the network. For regular users to front run others on the blockchain, they need to be fully/well connected to other nodes on the network. Doing so, they are able to listen to the network and monitor all transactions that are broadcast. On the Ethereum blockchain users have to pay for the computations in small amount of Ether called *gas* [?]. The price that users pay for transactions (a.k.a. transaction fees) can increase or decrease how quickly miners will execute them and include them within the block they mine. This is because the Ethereum miners consume resources to process the transactions and so receive the transaction fees after creating the blocks. Once seeing two identical transactions with different transaction fees, miners act to maximize their profit and are free to mine the transaction which offers the highest fee. Therefore, any regular users who run a full-node Ethereum client can modify the order of pending transactions by paying with a higher gas price*i.e.*, by monitoring the network, upon seeing a pending *buy* transaction which will further increase the price of the asset, a font-runner user can pay a higher gas price and hope to have his transaction a head of the other one mined. By doing so, he achieves a better rate from the other network users. Note that in this case, blockchain front-runners are more visible to the network as they broadcast their transactions to all the network participants.

4.2 Historical evidence in the Blockchain

As blockchain records are immutably recorded, there is enough historical data to analyze for possible front running detection. For examples here we research some of the events of such attacks happening in the Ethereum blockchain applications.

Status ICO

As discussed in ??, capped ICOs are a good application for front-running attacks. In June 2017, Status.im [?] started their ICO and within 3 hours they reached their cap, resulted in close to 300,000 Ether of funding. They used a *fair* token distribution method called *Dynamic ceiling*, which was in place to prevent large investments to take place early and limited the amount of Ether deposited, this was an attempt to increase the time window for smaller investors. They implemented multiple caps –ceiling– which each would have a maximum amount that can be deposited. In this case every deposit would be checked by the smart contract and the exceeding amount would be refunded to the sender

while the accepted amount would be sent to the Status.im wallet [?]. On the time of the ICO there were reports of Ethereum network being unusable and transactions were not confirming. Further study showed that some mining pools might have been manipulating the network for their own profit.

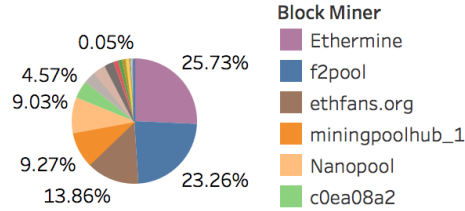


Fig. 1: The percentage of Ethereum blocks mined between block 3903900 and 3908029, this is the time frame in which Status.im ICO was running. This percentage roughly shows the hashing power ratio each miner had at that time.

F2Pool, an Ethereum mining pool that had around 23% of the mining hash rate at the time (Figure ??), sent 100 Ether to 30 new Ethereum addresses before Status.im ICO started. On the time the ICO started F2Pool constructed 31 transactions from the addresses they controlled to the ICO token sale smart contract –without broadcasting these transactions to the network– and focused their mining power to mine their own transactions. As shown in figure ??, most of the top miners in the mentioned time frame, have mined almost the same number of failed and successful transactions which were directed toward Status.im token sale, however F2Pool’s transactions indicate their successful transactions were equivalent to 10% of the failed transactions, hence maximizing the mining rewards, on gas, while censoring other transactions to the token sale smart contract.

Involvement of F2Pool in this action becomes obvious when the transactions from these 30 addresses are traced. As shown in figure ??, the funds deposited by F2Pool in these addresses were sent to Status.im ICO and mined by F2Pool. Once the dynamic ceiling algorithm refunded a portion of the sent funds back to these addresses, these funds were again sent back to F2Pool main address.

Although this incident does not involve transaction reordering in the blocks, it shows how miners can modify their code to behave in certain way to front-run other transactions and result in monetary profit.

Bancor

Bancor is an Ethereum-based application that allows users to exchange their tokens without any counter-party risk. This protocol aims to solve the cryptocurrency liquidity issue by introducing *Smart Tokens* [?]- ERC20 compatible tokens with a built-in liquidity mechanism that are always available to users. Smart Tokens can be bought and sold through the users smart contract at an

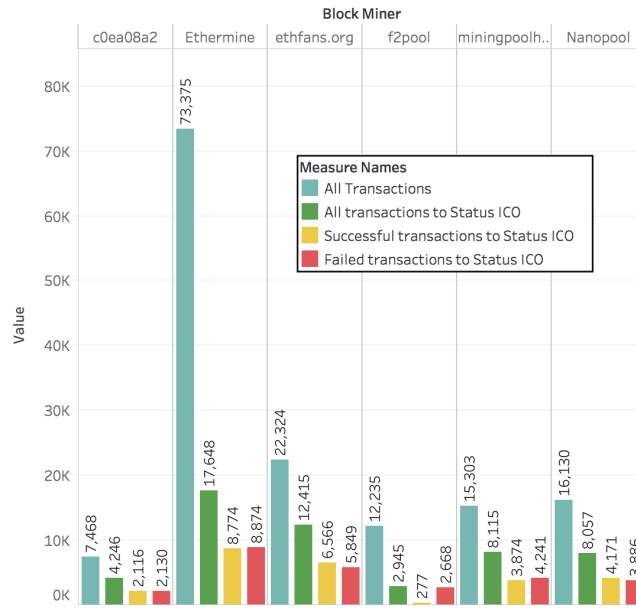


Fig. 2: This chart shows the miner behaviour on the time window that Status.im ICO was running.

automatically calculated price which displays supply and demand. Doing so, Bancor protocol provides continuous liquidity for digital assets without relying on an order book as there is no requirement to match sellers and buyers.

Front-running Bancor Recently, researchers have shown that Bancor is vulnerable to front running attacks. Implemented on the Ethereum blockchain, when Bancor transactions are broadcast to the network, they sit in a pending transaction pool known as *mempool* waiting for the miners to mine them. Since Bancor handles all the trades and exchanges on the Ethereum blockchain (unlike other existing decentralized exchanges), these transactions are all visible to the public for 16s (the average Ethereum blocktime) before being included within a block. This leaves this blockchain-based decentralized exchange vulnerable to the blockchain race condition attack as attackers are given enough time to front-run other transactions and gain favourable profits [?]. As mentioned in Section ?? and same as every other Ethereum based application, Bancor transactions can be front run by miners and other regular (non-miner) users. There is an implementation of a Bancor front running attack in the Python programming language which represent how a non-miner user can front run other network participants [?].

Domain Name Front-running

As mentioned in Section ??, although front-running attacks have been more showcased in the context of decentralized exchanges and trading systems, they

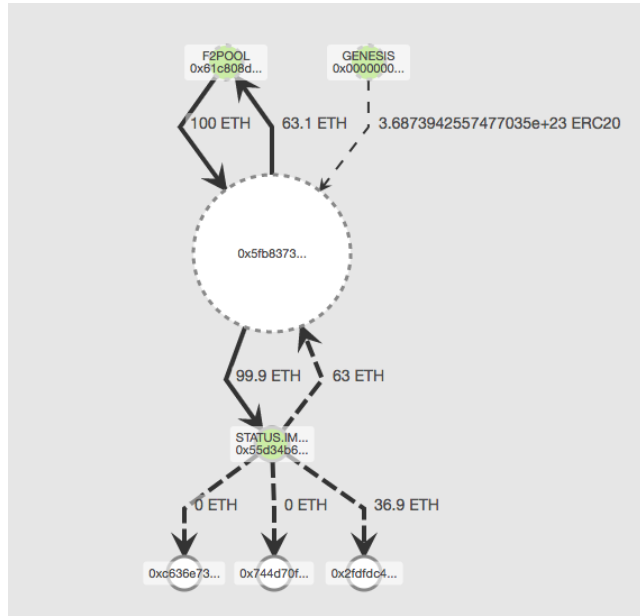


Fig. 3: F2Pool prior to Status.im ICO deposited 100 Ether in multiple new Ethereum addresses. On the time of the ICO they sent these deposits to Status ICO smart contract and prioritized mining of these transactions in their mining pool, this was to overcome the dynamic ceiling algorithm of the token sale smart contract. Later on they sent the refunded Ether back to their own address. [Graph made using Blockseer.com]

are are not yet limited to the financial markets. Front-running can occur within other (non-financial) blockchain applications such as naming systems. Blockchain-based domain registrar have been introduced to eliminate the role of central parties *i.e.*, domain name system (DNS) which introduce single point of failures in the entire web. One such system is Ghazal, an Ethereum-based naming and PKI system [?]. Ghazal users rely on the Ethereum blockchain to register their **.ghazal** domain names and bind certificates to those names. In Ghazal model, a user would register domain name by executing the *registerdomain* function from the Ghazal smart contract with the domain name in plain text as the function argument. As mentioned before, These transactions will sit in the mempool so that it would be mined by Ethereum miners and included in the block. During this period in which the transaction is not yet confirmed, front-running attack can occur by (i) dishonest miners and/or (ii) regular non-miner user. In the first case, a miner would intercept the *registerdomain* transaction and register that name ahead of the user. A regular non-miner node in the Ethereum network can front-run other user's *registerdomain* at a good profit by paying higher transaction fees. This practice allows the adversarial party to

register an unregistered domain immediately after someone searched for it, and sell the domain name to the users for a higher price. Note that, domain name front-running also occurs in the vanilla system. In this case, front-runners would monitor and identify domain names of interest, and sell it back to the other users with a higher price [?].

5 How to stop front-running?

In the traditional markets, front-running is considered unethical and also illegal. In the blockchain space, the designers of the decentral applications cannot rely on the justice system for unethical behaviour, and they must assume that each participant in the network acts rationally in their own self interest, the application will be operating in an adversarial environment [?]. A few decentral exchange projects such as *EtherDelta* and *0xProject* [?], have proposed solutions for front-running, which is off-chain orderbooks. The methods discussed within these projects prevent blockchain network participants from front running the orders as the orders are private and not known to the network and will not be broadcasted. However they are introducing third parties, *e.g.*, relayers in 0xproject, to be managing the orders with the promise not to reorder or front-run other orders.

Traditional front-running Prevention Methods. regulations/enforcement/broker education/sealed order

Blockchain front-running mitigation/prevention. The traditional methods of preventing front-running are based on regulation and restrictions applied to the brokers and actors within the markets, such methods do not apply to blockchains, no enforcement methods, ...

There are two main approaches to prevent front running, one to design a blockchain that is front-run-resistant, and the other to design the application logics in a way that front running is not possible.

5.1 front-run-resistant Blockchain

What does this mean to have a front-run-resistant blockchain? There are technical difficulties to achieve such solutions as there are unknown factors within such network designs (corner/edge cases).

Fixed transaction ordering: One possible solution to fix current blockchains in regards to transaction reordering by miners has been proposed by da Silva *et al.* [?], which consist of an algorithm, Fixed Transaction Ordering and Admission (FTOA) algorithm, in the consensus protocol that enforces the order of the transactions.

- **Privacy Preserving Blockchains:** Shielded transactions in ZCash do not reveal the sender, receiver, the amount and the data included in the transactions, hence they cannot be seen by network participants to be front-run. Although this limits the functionality of the blockchain/smartcontracts/-dapps
- **textbfLoopring/Dual Attesting:** Talk about this possible solution

5.2 Application Design to Prevent Front-running

Another way of preventing the front-running activities is to design the applications, whether they are decentralized (Dapps) or centralized, in such a way that no information would be available to the malicious parties which enables them to front-run the other users. Followings discuss the two techniques which can be used by the Apps/DApps developers while designing the systems to prevent front-running activities.

- **Commitment scheme:** Also known as *Commit and Reveal*, is a cryptographic primitive that enables one to commit to a value (*e.g.*, statement, document, *etc.*) while keeping it a secret and on a later time reveal the committed value [?]. Commitment scheme is a good method to prevent information leakage from the sensitive transactions. This could be done by simply broadcasting the hash value of the committed data and later on revealing the values that would be hashed to the committed hash. In the case of decentral exchanges, user can send a commit transaction which will be cryptic to network participants but will act as a placeholder in the queue for the user, after the transaction is mined, user sends reveal transaction revealing the order details which will be executed in a fair order. Another use of such commitment scheme, is in decentralized naming systems, such as Ghazal or ENS. In this case, a user would send a commit transaction similar to a sealed bid, once the transaction is confirmed and the grace period is over, user would send a reveal transaction revealing the bid and also the details of the requested domain. Using this scheme, one is able to hide information from the adversarial parties in the system and prevent them from front-running.
- **Submarine Commitments:** describe. General solution similar to commit and reveal. This is to solve the participatory factor of the commit and reveal solution.
- **Application logic specific solution:** Depending on the use case of the Dapp or the application, it could be designed in a way to de incentives some actors to front-run the transactions or prevent them from doing so. As an example on designing a decentral orderbook, it could be said that a miner could front run orders for financial gain on price improvement, however if the fees of the orders are sent to the miner of the block, it de incentives the miner to front run the orders as they already gain enough financial benefit from including the correct order of transactions. Another example for decentral orderbook design could be using call market design instead of time-sensitive orderbooks. In such design the arrival time of the order does not matter as they are executed in batches.

6 Concluding Remarks