Combating Front-running in the Blockchain Ecosystem

by

Jack Byers

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Computer Science

Lehigh University

January 2022

This thesis is accepted and approved in partial fulfillment of the requirements for the Master of Science.

Jack Byers

Combating Front-running in the Blockchain Ecosystem

_____

Date

_____

Thesis Advisor

_____

Chairperson of Department

Acknowledgements

I would first like to thank the members of the Computer Science & Engineering department at Lehigh who guided me through my life as a graduate student. Thank you to the members of the Scalable Systems & Software Research Group, I truly appreciate the opportunity to work and learn with you all. A particularly special thanks to Dr. Korth for his role as my advisor for the past five years, I would not be the student I am today without your guidance and knowledge.

Thank you to my family and friends, who supported me throughout my time at Lehigh. To my parents, I am endlessly grateful for everything you've done. Your encouragement and belief in me was unwavering and it inspires me every day.

Contents

Combating Front-running in the Blockchain Ecosystem

Jack Byers

December 5, 2021

Abstract

Front-running is generally defined as trading upon advance knowledge of some large transaction that will influence the value of an underlying security [1]. Due to the centralized nature of traditional financial markets, it is widely prohibited and its use eliminated by the threat of legal action. In the decentralized, trustless context of a blockchain, however, this approach is not available. This work seeks to generalize and quantify the problem of front-running, evaluate a number of proposed solutions, and contribute a framework for creating a front-running resilient blockchain system. Solutions are evaluated with particular respect to the principles and guiding goals of decentralized blockchain communities, and thus Bitcoin [2] and Ethereum [3] are used largely as examples throughout.

1 Introduction

The cryptocurrency industry has experienced drastic growth in recent years. At the time of writing, cryptocurrency exchanges handle upwards of $200 billion USD in daily volume, primarily concentrated in centralized exchanges [4]. These centralized exchanges largely do not have to contend with the issue of front-running, as they function using conventional order-book based settlement. This settlement is done privately, and transactions are submitted to centralized off-chain authorities such that front-running them is infeasible. While there are clear advantages towards centralized systems (transaction throughput, settlement time), they are generally seen as antithetical to the core values of decentralized blockchain communities. These centralized exchanges require trust that funds will be handled properly, and there are numerous examples of fraud, mismanagement, and collapse [5].

However, an alternative exists, and there is a growing movement towards *decentralized exchanges*, or DEXes. These DEXes are non-custodial by nature, and users are never required to give up control of their funds. Operational logic is written in smart contracts, whose code is public and whose execution can be publicly verified and audited. DEXes are largely transparent, atomic, and trustless. The downside, however, is that on-chain execution for most blockchains is slow. With settlement latency measured in dozens of seconds, malicious traders have ample time to analyze and execute complex transaction schemes to extract profit, leaving naive users damaged and the

network-at-large negatively impacted [6]. This is generally how the problem of front-running presents itself in the context of a blockchain ecosystem. Note that this work assumes the reader has a general familiarity with blockchain concepts as described in [7].

2 Background

Defined below are a number of concepts that are useful in understanding front-running in the blockchain ecosystem.

Decentralized Finance (DeFi) Decentralized finance generally refers to the collection of decentralized financial platforms that look to replicate the function of their centralized counterparts. This includes loaning services, exchanges, leveraging services, etc.

Decentralized Exchanges (DEXes) Decentralized exchanges such as Uniswap [8] and Sushiswap [9] are non-custodial platforms that facilitate trades directly between users. As opposed to a centralized exchange, where depositors must trust the platform to hold their funds safely, decentralized exchanges are entirely trustless. On Ethereum, decentralized exchanges are deployed as smart contracts and thus are inherently transparent and auditable.

Automated Market Maker (AMM) An automated market maker is the autonomous business logic used by DEXes to facilitate the pricing of

assets without the use of an order book. For centralized exchanges, the use of an order book to match buy and sell orders is logical, as the parties involved have already agreed to deposit their assets with the exchange. In a decentralized context, however, AMMs allow users to deposit assets into a "pool", from which they are logically guaranteed to be able to withdraw at any time. This pool usually consists of a pairing of two assets, such as USDC (a stablecoin whose value is pegged to the US Dollar) and ETH. Price is then determined by the ratio between those two assets in the pool. A user looking to exchange between the two assets can deposit one and withdraw an equivalent amount of the other (based on the resulting ratio) in a single atomic transaction.

Smart Contracts Smart contracts are programs whose compiled bytecode is stored on the blockchain. Execution of smart contract code is done as part of the consensus process, and thus the validity of their results is guaranteed by the network [10]. They can read and write to the state of the blockchain, and are fully public and immutable. Smart contracts have a number of applications including smart wallets, voting applications, exchanges, virtual casinos, video games, and more.

Transactions A transaction in the blockchain context is a set of instructions to translate the state of the system from one point to another. In the case of Bitcoin, these instructions are limited, and thus transactions are limited to debit/credit operations to move funds from one account

to another. However, on Ethereum and many other modern networks, transactions can invoke a broader range of actions by allowing *data* to be submitted. This data can then be processed on-chain by smart contracts, which enables complicated business logic to be executed in a trustless environment. Naturally, this entails a more complicated fee system, which in the case of Ethereum is addressed by *Gas*.

Gas Gas is the fee that is paid to execute a transaction on the ethereum network. Each computational operation has an equivalent cost in units of gas, and each user determines how much they are willing to pay per unit. Miners, in turn, prefer transactions that have the highest gas prices, and an economy is created in which users pay a network-average gas price per unit of computation for their transaction when submitting. The purpose of this system is to provide a bound on the amount of work that can be done by a single transaction, or be done in a single block. This way, scenarios like infinite loops or computationally intensive operations are prevented from slowing the network down. Unlike Ethereum, Bitcoin does not provide a Turing complete platform, therefore this is not a consideration for that network. A side effect of this system, however, is that users can roughly control the ordering of their transaction in a block by strategically setting their gas price below or above another transaction.

Front-running As defined in the abstract, front-running is trading upon ad-

vance knowledge of some large transaction that will influence the value of an underlying security [1]. In the specific context of the blockchain ecosystem, front-running entails monitoring the public transaction pool for transactions that either generate value for their submitter directly, or can be used to generate value (most commonly via arbitrage). When such a transaction is found, a front-runner will attempt to submit transactions that either duplicate or profit off of the target transaction. By carefully manipulating placement within the final ordering through transaction fee auctions [11] or network spam, the front-runner can guarantee themselves profit.

Back-running Back-running specifically refers to an adversarial attack strategy where a transaction is placed directly behind another. This can involve packing a block with transactions at a gas price just lower than the target transaction. For example, when a price oracle publishes a price update to the network, it will affect the pricing on many exchanges. Miners make seek to back-run this transaction so they are the first to act on its information.

Flash Loans Flash loans are atomic loans that can be withdrawn from the decentralized lending service, and paid back within the same block. Since this transaction happens atomically, the *value of the loan is only limited only by the holdings of the lending service.* This allows traders to leverage the massive (and constantly growing) amount of value staked

in these lending services. While not inherently a malicious tool, it effectively lowers the barrier of entry to many attack types. This means that a user can open a flash loan, use it to manipulate the price of a currency and create an arbitrage opportunity, then execute on that opportunity, pocket the profit, and return the loan all in a single block. The effect and capability of flash loans is difficult to predict, and has gone largely unstudied [12].

Miner Extractable Value (MEV) Miner Extractable Value is the value miners get, on top of block rewards, as a result of front-running and gas price auctions [13]. This value is received from users in the form of inflated gas fees, which are a result of transaction ordering auctions plaguing the ethereum network. MEV is largely advantageous for the miners but it poses a threat to blockchain security. Particularly of concern is the possibility for miners to be financially incentivized to reorganize the blockchain and claim previous blocks' MEV. As the rewards could potentially be higher than the slashing penalty, this is a serious security concern. Researchers have started using the term Blockchain Extractable Value (BEV) instead of MEV to include the value taken by other entities such as trading bots. It is important to note that miners have ultimate control over the placement and ordering of transactions within a block. Miners can harvest a large amount of value by running specialized clients that optimize for this opportunity. If mass adoption of such a strategy occurs, it could disrupt fundamental economic as-

sumptions of these networks, and potentially impact their safety from rewriting past blockchain events [14], [15].

3 Purpose

The effects of front-running activity are felt throughout the network. In the case of Ethereum, transaction ordering in a block is impacted by the gas fee one is willing to pay a miner per unit of work done. This fee is controlled by the transaction submitter, and thus transaction order can be influenced through manipulation of this fee. A reasonable miner who is not optimizing transactions for themselves will order transactions by the amount paid for each unit of gas. When a valuable opportunity is discovered by users, it will often result in real-time bidding wars for preferable transaction ordering [13]. These MEV opportunities drive up gas costs for the entire network, as computation becomes a premium. In times of severe volatility, arbitrage becomes more prevalent, and gas fees rise drastically. This problem is not unique to Ethereum, as all blockchain networks have had to contend with what is fundamentally the issue of order fairness.

While consensus is largely concerned with the properties of consistency and liveness, these networks are now contending with a third property: ordering [16]. The implications of transaction ordering in a blockchain context extend beyond scenarios of front-running and arbitrage. These issues fundamentally represent a vulnerability in the consensus protocol of both Bitcoin

and Ethereum [14], [15]. As the economic value held by these ecosystems grows, so does the incentive to exploit them for profit. It is estimated that a massive portion of this MEV can be extracted simply by re-ordering past transactions in the historical chain [13]. If a sufficient portion of mining power recognizes this value, then a re-ordering could occur which violates the immutability property for which public blockchains are considered valuable.

Additionally, not every impact of MEV is inherently harmful. In some cases, the valuable opportunities a protocol generates incentivize users to take actions that benefit the health of that protocol. For example, numerous DEXes use an AMM-based strategy to value assets, and these trading pools are kept in line through efficient and timely arbitrage. When one asset is exchanged for another in a pool, the relative value of the original asset is increased and an arbitrage opportunity is presented. These decentralized exchanges rely on traders identifying this opportunity as quickly as possible, and executing trades such that the ratio of assets in the pool is in line with other exchanges. In this case, the ecosystem is relying on the presence of public information and the behavior it encourages. Note that this explicitly does not include the aspects of MEV that threaten the chain with re-organization, and only considers the situation from the perspective of a DEX. At a larger scale, MEV opportunities across the entire protocol can create value considerable enough to threaten the security of the network, and thus the problem should be addressed with care.

4 Solutions

This section covers a number of proposed solutions to the front-running problem, specifically on Ethereum. While Ethereum is by no means the only blockchain ecosystem where front-running is prevalent, it has the largest DeFi market and thus is the target of the majority of malicious activity. The smart contract capability enables users to construct far more complex and powerful trading strategies, and thus the opportunities for profit are more numerous in this context. Accordingly, it is also the environment for which solution development is most active.

To assist with the evaluation of these solutions, we propose three categories that solutions generally fall into: *Exchange-Level*, *Transaction-Level*, and *Consensus-Level* solutions.

4.1 Exchange-Level Solutions

Exchange-level solutions are those that aim to address front-running at the DEX level (or, as it is known in the Ethereum community, layer-2/L2). These solutions are entire DeFi platforms that handle transactions in such a way that their content is not revealed until after they have been committed to the chain. Obviously, this does not address front-running in such a way as to eliminate it entirely, however, if front-running-resilient exchanges grew in popularity it would likely have positive impacts on the health of the ecosystem.

One example of such a platform is TEX - A Securely Scalable Trustless Exchange [17]. TEX provides front-running resilience using a series of clever cryptography techniques. Named the *moonwalk proof*, it is essentially a time-lock puzzle combined with a zero-knowledge proof that the transaction is syntactically valid. Therefore, the transaction can be accepted onto the chain but is cryptographically guaranteed to not have its details revealed until after the next block has been committed. If a user suspects that they have been front-run, they can issue a challenge to the network, which will evaluate whether or not the challenged party withheld information to break the commit-reveal scheme. If this is the case, the transaction is reverted and funds are returned.

There are two key downsides to this approach. The first is that the burden of proving that front-running occurred is placed on the user. Each user must monitor their own trades to ensure they haven't been front-run, and this creates a barrier to entry on the blockchain ecosystem. Second, this process is cryptographically intensive, and would generate a large amount of computational load in the form of transaction fees.

4.2 Transaction-Level Solutions

Transaction-level solutions are those that propose a generalized scheme to submit transactions in such a way that they cannot be front-run. Typically, this either involves trusting an additional party, or incurring a considerable additional delay or cost. However, since they can generally apply to any

transaction, there is a greater scope of implied benefit from mass adoption of such a strategy.

One example of such a solution is LibSubmarine, and their submarine-send protocol. The strategy they employ is a smart contract that will reveal the details of a transaction after it has been committed, once that smart contract is provided with the appropriate private key. It is a simple encryption scheme, with a secondary transaction fee on top of the existing one. It also requires DeFi applications to support sends through the submarine-send protocol.

Another, more straightforward example is simply developing a relationship with a group of miners you trust. While this may sound silly, it's the heart of the MEV-Auction Relay system that has recently gained some popular adoption [18]. Essentially, instead of submitting your transaction to the public pool, where miners can reorder it at will, you submit it to a relay network that bundles it with other transactions. This bundle is then passed to a trusted list of miners who guarantee that it will be placed at the top of a block. While you do need to include a tip for these miners to ensure it is economically reasonable for them to place your transactions at the top, there is guaranteed to be no gas-price-based bidding war, as the transactions are bundled.

4.3 Consensus-Level Solutions

Consensus-level solutions are solutions that aim to address the front-running problem from a fundamental consensus perspective. Such a solution would change the underlying consensus process to avoid the issue of order fairness altogether. However, this becomes a difficult task, since you need to define *fair*. Should a user be able to pay to have their transaction placed higher? If you choose to sort transactions by the average timestamp at which they were received, then users with poor connection (perhaps, those in remote areas of the world) will almost always be placed closer to the bottom of a block. However, this isn't necessarily an issue, as most users only care about per-block ordering when attempting to front-run other transactions. Still, determining what exactly is "fair" is a difficult decision. It is the opinion of the author that reaching consensus on an ordering of transactions, based on their arrival time to the network, is the best possible solution. Networks should encourage users to be as well-connected to other peers as they can be.

One example of a non-arrival-time based solution is a modified version of the go-ethereum client released by the Flashbots team [19]. For each block, this client runs an auction, where individual miners compete for the right to order a block. This way, individual traders lose their ability to control ordering in the block, and miners are instead compensated by reaping the full value of available MEV. While Avalanche [20] reduces the effects of front-running at the consensus level, they accomplish this by privatizing the transaction pool to validators until the block is finalized. The result is a

system that, while transaction fees are not impacted, is still aggressively exploited for MEV rewards.

5 Contribution

The following section provides clarification on the desirable qualities of a solution to the front-running problem, along with contributing an approach that could be used to combat front-running on the layer-1 level.

5.1 Motivation

The solutions covered in the previous section generally all compromise the values of blockchains networks in some fundamental fashion. Those that do not are typically too precise in scope and would fail to address the problem in its entirety. Additionally, solutions like TEX [17] suffer from relying on an exact definition of front-running that may prove insufficient over time. As blockchain networks grow and adapt, they may change their core structure in ways that open up capability to users that did not exist previously. Such changes (like Eth2 [21]) may provide users with tools that exacerbate the issue of front-running, as flash loans have done on Ethereum [12].

Thus, a need exists for a solution that does not have the shortcomings of existing proposals. A more promising approach may be to use randomness to reduce the probability that a user's ability to take malicious action is negligible. A number of popular blockchain networks (notably Algorand

[22]) have taken advantage of clever cryptographic tools to achieve desirable network properties. The proposed system draws upon these tools to combat front-running behavior by making it probabilistically non-viable.

5.2 Key Concepts

The core cryptographic concept at use in this system is the VRF, or Verifiable Random Function [23]. Used extensively in Algorand, VRFs can be used to map inputs to pseudorandom outputs (discussed in detail following this passage). There exist similar tools in a number of blockchain systems, such as RANDAO in Ethereum 2.0 [24].

Using the properties of VRFs, Algorand defines the notion of *cryptographic sortition*, which is used to securely and randomly distribute staked users into network roles. More importantly, this process can be done independently and privately, with network verification following its completion. Users calculate their input using a cryptographic key in addition to some shared entropy (such as the previous block time) and submit this to a VRF. If the output falls within a certain threshold, signifying a given role, they can then reveal their input to the network to "prove" that they were chosen for said role. There are a number of important considerations, such as the Sybil attack [25], but they are well defined and addressed within the original Algorand work [22]. Generally, weighting users' odds of selection in sortition by their amount of staked funds, in addition to slashing them for malicious behavior, makes foul play economically infeasible.

Additionally, the proposed system draws heavily from the work of Chainlink's Fair Sequencing Service [26], and related works in Byzantine order fairness (namely the *Aequitas* family of consensus protocols) [16]. The Fair Sequencing Service (or FSS) provides an interesting framework at the layer-2 level for building quorums around given transaction ordering among nodes.

5.3 System Structure

The structure of this network is a strict modification of the Algorand system as defined in its whitepaper. However, as *validators* in Algorand validate transactions without considering their ordering within the block, the concept of the *orderer* role is introduced. Staked users selected for this role are not allowed to submit transactions for the given block, and are given the explicit task of ordering a list of selected transactions, which are then passed off to the validators.

At the beginning of a block, transactions are gossiped publicly between all nodes on the network (including validators and orderers, however, they are not aware of their role yet). At the end of the epoch, sortition is run for staked users. Users submitting for the role of orderer must not have any transactions in the pool, or they will be slashed. After orderers are selected, they engage in quorum-building around ordered lists of transactions. This ordered list is constructed following the process defined by FSS [26]. Orderers log the receipt of every transaction during the gossiping phase, and transactions are then selected from that log of receipts. Then, using

a leaderless implementation of the Aequitas protocol, the network reaches consensus with the property that if a majority $n > 2/3$ of nodes received tx_A before tx_B, then tx_A will be placed before tx_B in the final ordering. Once a quorum has been reached on a given ordering, it is passed off to validators, which ensure that it is syntactically valid, otherwise the process repeats.

By separating the process of ordering from validation, the network probabilistically eliminates users from manipulating ordering to their advantage. Front-running in such a system would require collusion between a majority number of transaction orderers, which is deemed non-probable by the same economic guarantees and randomization that secure the validation process.

Additionally, the complexity of Aequitas could be removed from this system and replaced with a threshold encryption scheme [27], as suggested by Chainlink. In this case, an encryption scheme is developed wherein transactions are encrypted before being distributed for gossip in the public pool. Only after the close of the epoch and the completion of "ordering" are they decrypted for validation. While such an approach would increase transaction fees for the required computational cost of encryption, it prevents any reordering based on transaction data, as it will not be visible until ordering is finalized.

5.4 Concerns

One concern with the proposed system is that (particularly for the threshold encryption approach) it incentivizes spamming the transaction pool

with multiple copies of the same transaction in the hopes that one lands in a preferable place in the ordering. Considering the overhead from encryption, this may have adverse effects on the network if the economic rewards from winning a good placement outweigh the cost spent in gas fees. Ideally, the system should have the property that outside of how long it takes to communicate their transaction to their peers, the user has no way of controlling (probabilistic or otherwise) a transaction's placement within the block. As the system does not natively address this concern, special consideration should be taken to the implementation of transaction fees in the network, perhaps exponentially increasing fees for successive transactions with the same state translation.

Another point of concern is the computational intensity of the Aequitas protocols. As discussed in the Chainlink 2.0 whitepaper [28], current implementations introduce massive communication overhead to the network, which makes its use in any practical BFT scenario doubtful. However, there have already been proposed a number of more lightweight solutions, whose performance may be suitable for a high-throughput environment [29].

6 Future Work

While this paper provides a foundation to further the discussion of comprehensive solutions to the front-running problem, there are a number of areas in which it can be improved upon and extended.

Rollup-Based Exchanges As research progresses in the field of zero-knowledge, a number of DeFi services have launched that use cryptographic techniques like zksnarks to collect, prove, and submit layer-2 transactions to the main chain in a non-interactive fashion [30]. One such example is Loopring [31]. As Ethereum moves towards a sharded POS system and attempts to incentivize the use of layer-1 as a data-availability layer [32], it will be important to study how front-running targets these rollup-based exchanges.

Algorand Implementation With Algorand's open-source implementation widely available via Github [33], it would be a valuable study to implement the *orderer* role on top of the existing Algorand network. This way, various ordering consensus solutions could be benchmarked to determine their effects on the cost of consensus.

7 Conclusion

This paper began by outlining the problem of front-running and the danger it poses to modern blockchain networks. This problem extends beyond simple arbitrage opportunities, and is part of a larger classification of Miner-Extractable Value, whose implications threaten the fundamental structure and security guarantees of various protocols. This work provided a collection of background information to assist in better defining the issue, along with a series of proposed categories that can be used to evaluate solutions.

Additionally, this paper covered a number of systems that aim to address front-running and evaluated them on their ability to resolve the issue.

Finally, this paper outlined a system that could properly combat the issue of ordering transactions, thus negating the effects of MEV and front-running. This paper does not define complete network specifics for such a system, but instead summarizes how such a system could be constructed using modern tools from related works. Overall, the author hopes that the definition and frameworks defined here can be used to further the conversation on finding a proper, long-term solution to front-running and its impacts on the blockchain ecosystem.

8 References

[1] *Front-running; an Unethical Behavior*, Sri Lanka SEC, Sep. 2012. [Online]. Available: http://www.cmic.sec.gov.lk/wp-content/uploads/2012/09/Front-running.doc-an-Unethical-Behavior.pdf.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Tech. Rep., 2009. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf.

[3] V. Buterin, "Ethereum white paper: A next generation smart contract & decentralized application platform," 2013. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper.

[4] "Cryptocurrency Global Charts," CoinGecko. (2021), [Online]. Available: https://www.coingecko.com/en/global_charts.

[5] C. Doughtery and G. Huang, "Mt. Gox Seeks Bankruptcy After $480 Million Bitcoin Loss," Feb. 2014. [Online]. Available: https://www.bloomberg.com/news/articles/2014-02-28/mt-gox-exchange-files-for-bankruptcy.

[6] "Ethereum Average Block Time," Etherscan. (2021), [Online]. Available: https://etherscan.io/chart/blocktime.

[7] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database System Concepts*, 7th. McGraw Hill, 2020, ch. 26, Blockchain Databases.

[8] "Uniswap," Uniswap. (2021), [Online]. Available: https://uniswap.org/.

[9] "Sushiswap," Sushiswap. (2021), [Online]. Available: https://sushi.com/.

[10] N. Szabo, "Formalizing and Securing Relationships on Public Networks," 2021. [Online]. Available: https://firstmonday.org/ojs/index.php/fm/article/view/548/469.

[11] E. Felten, "Mev auctions considered harmful," Medium, Tech. Rep., 2020. [Online]. Available: https://medium.com/offchainlabs/mev-auctions-considered-harmful-fa72f61a40ea.

[12] K. Qin, L. Zhou, B. Livshits, and A. Gervais, "Attacking the defi ecosystem with flash loans for fun and profit," *CoRR*, vol. abs/2003.03810, 2020. [Online]. Available: https://arxiv.org/abs/2003.03810.

[13] P. Daian, S. Goldfeder, T. Kell, *et al.*, "Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges," *CoRR*, vol. abs/1904.05234, 2019. arXiv: 1904.05234. [Online]. Available: http://arxiv.org/abs/1904.05234.

[14] M. Carlsten, H. Kalodner, A. Narayanan, and S. M. Weinberg, "On the instability of bitcoin without the block reward," *ACM CCS*, pp. 154–167, Oct. 2016.

[15] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," Cornell University, Tech. Rep., 2014. [Online]. Available: https://arxiv.org/pdf/1311.0243.pdf.

[16] M. Kelkar, F. Zhang, S. Goldfeder, and A. Juels, "Order-fairness for byzantine consensus," Tech. Rep., 2020. [Online]. Available: https://ia.cr/2020/269.

[17] R. Khalil, A. Gervais, and G. Felley, "Tex - a securely scalable trustless exchange," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 17, 2019. [Online]. Available: https://ia.cr/2019/265.

[18] "Flashbots relay." (2021), [Online]. Available: https://docs.flashbots.net/flashbots-protect/overview.

[19] "Flashbots auction." (2021), [Online]. Available: https://docs.flashbots.net/flashbots-auction/overview/.

[20] T. Rocket, M. Yin, K. Sekniqi, R. van Renesse, and E. G. Sirer, "Scalable and probabilistic leaderless BFT consensus through metastability," *CoRR*, vol. abs/1906.08936, 2019. [Online]. Available: http://arxiv.org/abs/1906.08936.

[21] "Eth2 upgrades," Ethereum. (2021), [Online]. Available: https://ethereum.org/en/eth2/.

[22] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," Shanghai, China,

Tech. Rep., 2017, pp. 51–68. DOI: 10.1145/3132747.3132757. [Online]. Available: https://doi.org/10.1145/3132747.3132757.

[23] M. Silvio, R. Michael, and V. Salil, "Verifiable random functions," *In Proceedings of the 40th Annual Symposium on the Foundations of Computer Science (FOCS '99)*, 1999.

[24] Y. Qian, *Randao*, https://github.com/randao/randao, 2015.

[25] J. R. Douceur, "The sybil attack," in *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, London, UK: Springer-Verlag, 2002, pp. 251–260, ISBN: 3540441794. [Online]. Available: http://portal.acm.org/citation.cfm?id=687813.

[26] A. Juels, L. Breidenbach, and F. Tramèr, "Fair sequencing services: Enabling a provably fair defi ecosystem," Chainlink Blog, Tech. Rep., 2020. [Online]. Available: https://blog.chain.link/chainlink-fair-sequencing-services-enabling-a-provably-fair-defi-ecosystem/.

[27] C. Cachin, K. Kursawe, F. Petzold, and V. Shoup, "Secure and efficient asynchronous broadcast protocols," in *Advances in Cryptology — CRYPTO 2001*, J. Kilian, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 524–541, ISBN: 978-3-540-44647-7.

[28] L. Breidenbach, C. Cachin, B. Chan, *et al.*, "Chainlink 2.0: Next steps in the evolution of decentralized oracle networks," Tech. Rep., 2021.

[Online]. Available: https://research.chain.link/whitepaper-v2.pdf.

[29] K. Kursawe, "Wendy, the good little fairness widget," *CoRR*, 2020. [Online]. Available: https://arxiv.org/abs/2007.08303.

[30] "What are zk-snarks?" (2021), [Online]. Available: https://z.cash/technology/zksnarks/.

[31] W. Daniel, Z. Jay, W. Alex, and F. Matthew, "Loopring: A decentralized token exchange protocol," 2018. [Online]. Available: https://loopring.org/resources/en_whitepaper.pdf.

[32] V. Buterin. "An incomplete guide to rollups," vitalik.ca. (2021), [Online]. Available: https://vitalik.ca/general/2021/01/05/rollup.html.

[33] "Algorand's official implementation in go." (2021), [Online]. Available: https://github.com/algorand/go-algorand.

9 Vita

Jack Byers grew up in Newburyport, Massachusetts, where he lived with his parents Tracy Byers and Brent Byers. Jack attended Lehigh University as an undergraduate student, majoring in Computer Science and Business and graduating with High Honors in January 2021. As an undergraduate student, Jack worked as a Grader for various undergraduate Computer Science courses, and was a P.C. Rossin Engineering Junior Fellow. In this role, Jack participated in various leadership duties including community outreach activities, attending panels to answer questions for prospective engineering students, and giving campus tours. Jack was awarded the Presidential Scholarship from Lehigh, and subsequently enrolled in the Master of Science in Computer Science program at Lehigh University. He began this program in January 2021, and graduated a year later following the Fall 2021 semester. As a graduate student, Jack worked as a Graduate Assistant for the CSE297: Blockchain Algorithms & Systems course. Additionally, Jack worked in the Scalable Systems & Software Research Group at Lehigh, studying under Professor Hank Korth. Following graduating, Jack intends to work as a Software Engineer in the distributed systems field.