Behind Block Explorers: Public Blockchain Measurement and Security Implication

Hwanjo Heo
KAIST/ETRI
Daejeon, South Korea
hwanjo@kaist.ac.kr

Seungwon Shin
KAIST
Daejeon, South Korea
claude@kaist.ac.kr

***Abstract*—Blockchain data has become a popular subject in studying various aspects of blockchains including the security of underlying mechanisms. However, the main chain block data, usually available from block explorer services, does not serve as a sufficient source of transaction and block dynamics that are only visible from a large-scale event measurement. In this paper, the transaction and block arrival events of the two popular public blockchains, i.e., Bitcoin and Ethereum, are measured to investigate the hidden dynamics of blockchain networks. We share our key findings and security implications including a false universal assumption of previous mining related studies and an invalid transaction propagation problem that can be exploited to launch a Denial-of-Service attack on a network.**

I. Introduction

Since the inception of Bitcoin – the first peer-to-peer distributed ledger system invented by Satoshi Nakamoto [36] – in 2009, many blockchain systems have undergone development in the public. Ethereum [9] has started its mainnet in 2015 to enhance the vision of blockchain by featuring the idea of smart contracts; it is recognized as the first decentralized computing platform for decentralized applications (dApps). Public blockchains have evolved afterwards to provide better privacy [32], scalability [3], and financial service specialization [42].

Information recorded in blockchains, beyond their original utility of being transaction ledgers, are publicly available and commonly studied in the literature. For example, block explorer services [4], [5], [7], [8], [17] have emerged to provide transaction information stored in blockchains with additional analytics and user-friendly interfaces. Similarly, historical block and transaction data have become a popular subject for not only economic aspects of cryptocurrencies such as cryptocurrency abuse [29], anti-money laundering [34], and monetary flow of businesses [30], but also to evaluate the security of underlying mechanisms of blockchains [22], [23].

However, *we claim that historical blockchain data is not sufficient* to evaluate many aspects of blockchain systems such as blockchain forks, mining pool behavior, and revealing potential DoS vectors in blockchain nodes; live measurement of block and transaction arrival events is an indispensable source that provides the full visibility of block and transaction dynamics. Block explorer services, or simply querying a locally stored blockchain database, that are popularly used in previous studies, do not provide much richness in the way of block and transaction information besides what is recorded in the main chain blocks.[1]

In this paper, we conduct in-depth analysis of block and transaction arrival event measurement of the two popular blockchains: Bitcoin and Ethereum. Table I summarizes the two blockchains we study in this paper. The block and transaction dynamics, which are not visible from the main chain block data, are investigated to reveal our findings and their consequent implications. Specifically, we make the following contributions:

- Transaction and block arrival events of the two major public blockchains are measured in eight geographically dispersed locations. We perform a quantitative analysis on the discrepancy in observed transactions and blocks to identify yet unveiled dynamics and security implications that are not apparent from historical blockchain records.
- We study blockchain forks to show our findings including the universal assumption adopted by mining related research stating that honest miners add blocks to the earliest propagated block in the event of blockchain forks is not the case.
- We show that the time gaps between the block arrival and the latest transaction arrival of the said block, are not small. We further examine the mechanism behind the gaps to show how severely a miner's reward is affected.
- We identify a subtle security problem of Ethereum that is attributable to its account model. The vulnerability we discover can be further exploited by an attacker to launch a network-wide Denial-of-Service (DoS) attack.

II. Motivation & Research Questions

Our motivation for this measurement study comes from the fact that historical blockchain data, that is locally stored by full node software processes or provided by block explorer services, do not reproduce dynamic transaction and block events observed at blockchain nodes. Figure 1 describes an illustrative timeseries of transaction and block arrival events in two different view models; historical blockchain data only provides a *historical view* in which transactions are flattened, i.e., appear as if simultaneously arriving at the time of block arrival (Section VI), and only those included in the main

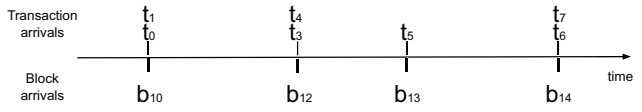
[1]See Section VI for our block explorer survey result.

2575-8411/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDCS51616.2021.00029

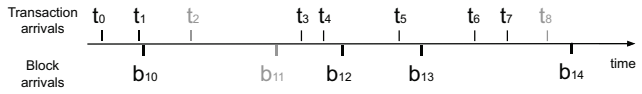
TABLE I: Blockchain summary

Blockchain	Consensus mechanism	Block time	Transactions per second (all time high)	Balance model	Transaction fee	Peer-to-peer management		
						Discovery	Max # peers	(outgoing)
Bitcoin	PoW	∼10 minutes	7	UTXO	Yes	Random from seeder list	125	(8)
Ethereum	PoW	∼10 seconds	15	Account	Yes	Kademlia-like	50	(16)

chain blocks are visible. Transactions and blocks can be stale, i.e., not included in blocks or not part of the main chain respectively, or not even explicitly advertised if observed under the hood. A *measurement view* can provide a rich and dynamic transaction and block arrival event logs, on the other hand.



(a) Historical view



(b) Measurement view

Fig. 1: Two view models of blockchain events; grey-colored events are only observable in the measurement view.

Our study aims to shed light on the understanding of various aspects of blockchains obscured from a historical view. Specifically, we have focused on three research questions:

RQ1: Are network-advertised transaction logs consistent with the transactions included in main chain blocks?

- (preview on our findings) No. There are unadvertised transactions that are directly inserted by miners, implying privatization of transaction fees (Section III-C2). Advertised transactions also may end up being stale, i.e., not included in blocks, mostly due to conflict transactions which instead get placed in blocks (Section IV-C); a particular kind of stale transactions, i.e., insufficiently funded transactions, can be further exploited to launch a DoS attack (Section IV-D).

RQ2: Do miners behave rationally (or selfishly) to maximize their profit? (Do miners endorse earliest propagated blocks in the event of blockchain forks? Do mining pools faithfully maximize their profit in terms of the transaction fee?)

- (preview on our findings) No. Bitcoin fork events now have become too rare (0.04%), implying that selfish mining is not the preferred strategy for Bitcoin miners. On the other hand, the majority of Ethereum miners add blocks to their own mined blocks rather than the earliest propagated ones that are mined by other pools; this contradicts the unarguably adopted assumption of previous studies (Section IV-A). Furthermore, Bitcoin and Ethereum mining pools give up 5.15% and 40.39% of transaction fee rewards, respectively, for their mined blocks due to sluggish block proposal update (Section IV-B).

RQ3: Do blockchain nodes effectively get away from partitioning attacks by virtue of information redundancy from multiple peers? (Are they still vulnerable to partitioning attacks? Is there any side-effect of maintaining a large number of peers as a countermeasure?)

- (preview on our findings) Yes. The reference implementations of PoW blockchains have increased the number of (outbound) peer connections and, consequently, the up-to-date nodes benefit from redundant block advertisement from 14 and 46 peers for Bitcoin and Ethereum, respectively (Section V-1); tampering with a small number of peers will not successfully partition or make a node unsynchronized for minutes [24]. On the other hand, making a large number of peer connections can easily be circumvented by an adversary exploiting transaction ambiguity (Section IV-D).

III. Measurement

A. Setup

We have modified the reference implementations of two popular blockchain softwares – Bitcoin Core [12] and Go Ethereum [16] – to facilitate logging of all received transactions and blocks from peers. Our monitoring softwares are deployed to eight geographically dispersed regions of the Google cloud platform [25]: Asia-East, Asia-South, Europe-North, Europe-West, Northamerica-Northeast, US-Central, US-East, and US-West. Each VM is configured to have two vCPUs, 18.5 GB of memory, and 800 GB of standard persistent disk. Measurements were performed between July 2019 and December 2019 by having our monitors, which are indeed full nodes on the two blockchain mainnets, being connected to the peers. Unfortunately, we experienced several problems that resulted suspension of our monitoring processes. For example, the Google cloud platform endeavors to detect cryptocurrency mining related activities. Once detected, the VM instance immediately gets suspended and reinstated only if a reasonable explanation is provided.

B. Analysis Model

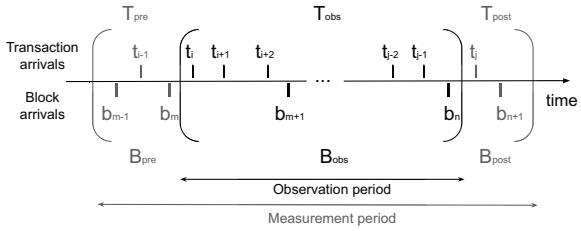


Fig. 2: Transaction and block analysis model; grey-colored events are cropped out for measurement point consistency.

TABLE II: Transaction and block set notation of our measurement framework

Notation	Description	Definition
B_{obs}	The set of blocks observed in the observation period	$\{b_i, b_{i+1}, \ldots, b_{j-2}, b_{j-1}\}$ where there is no b_k satisfying $\mathsf{arrival}(b_m) \leq \mathsf{arrival}(b_k) \leq \mathsf{arrival}(b_i)$ or $\mathsf{arrival}(b_{j-1}) \leq \mathsf{arrival}(b_k) \leq \mathsf{arrival}(b_n)$
B_{main}	The set of blocks that are included in the main chain	$\{b_{m+1}, b_{m+2}, \ldots, b_{n-1}, b_n\}$ where $b_i = \mathsf{prev}(b_{i+1})$ for all $m < i < n$ and b_m, b_n are the preamble and concluding blocks, respectively.
B_{stale}	The set of observed blocks that are stale	$B_{obs} - B_{main}$
B_{unseen}	The set of blocks that are part of the main chain but not observed	$B_{main} - B_{obs}$
T_{obs}	The set of transactions observed in the observation period	$\{t_i, t_{i+1}, \ldots, t_{j-2}, t_{j-1}\}$ where there is no t_k satisfying $\mathsf{arrival}(b_m) \leq \mathsf{arrival}(t_k) \leq \mathsf{arrival}(t_i)$ or $\mathsf{arrival}(t_{j-1}) \leq \mathsf{arrival}(t_k) \leq \mathsf{arrival}(b_n)$
T_{pre}	The set of transactions observed before the observation period	$\{t_0, t_1, \ldots, t_{i-2}, t_{i-1}\}$ where there is no t_k satisfying $\mathsf{arrival}(t_{i-1}) \leq \mathsf{arrival}(t_k) \leq \mathsf{arrival}(b_m)$
T_{post}	The set of transactions observed after the observation period	$\{t_j, t_{j+1}, \ldots\}$ where there is no t_k satisfying $\mathsf{arrival}(b_n) \leq \mathsf{arrival}(t_k) \leq \mathsf{arrival}(t_j)$
T_{stale}	The set of observed transactions that are stale	$T_{obs} - \bigcup_{b \in B_{main}} \mathsf{transactions}(b)$
T_{unseen}	The set of transactions that are included in the main chain blocks, but not observed	$\bigcup_{b \in B_{main}} \mathsf{transactions}(b) - T_{pre} \cup T_{obs} \cup T_{post}$

We briefly describe the blockchain model which generalizes the two public blockchains we are studying in this paper:[2]

- Blocks are produced by miners and are broadcast to the network by peer connections. A block b is uniquely identified with its *blockhash* and refers to its parent block b_p (i.e., $\mathsf{prev}(b) = b_p$). The arrival time of b at a measurement point m (i.e., $\mathsf{arrival}(b)^m$) is the arrival time of b from one of m's peers who delivers b for the first time. The block b can be included in the main chain[3] (i.e., B_{main}), or it can be stale (i.e., $b \notin B_{main}$).
- Transactions are composed and propagated by users through a P2P network. A transaction t is uniquely identified with its *txid*. The arrival time of transaction t at a node m (i.e., $\mathsf{arrival}(t)^m$) is the arrival time of t from one of m's peers who delivers t for the first time. The transaction t can be included in block b (i.e., $t \in \mathsf{transactions}(b)$) or being stale (i.e., $t \notin \mathsf{transactions}(b)\, \forall b \in B_{main}$).

Our model comprises the two arrival events, i.e., block and transaction arrival events, where a single block arrival event is mapped to zero or more transaction events (i.e., $t_i \in \mathsf{transactions}(b)$) by the function $\mathsf{transactions}(\cdot)$. Figure 2 depicts how we analyze our measurements. In studying the discrepancy between the transaction and block arrivals (Section IV), the arrival events near the measurement boundaries significantly affect our quantitative analysis. For example, including or excluding a single Bitcoin block around the measurement boundary can produce more than two thousand stale or unobserved transactions. To this end, we crop the measurement to the *observation period* by carefully choosing two anchor blocks, the preamble block b_m and the concluding block b_n, for each data set of blockchains so that our measurement satisfies the two conditions: (i) there is no loss of measurement data (due to the reasons described in Section III-A) at any of our eight locations in the time period of one week before the arrival of b_m and one week after the arrival of b_n, and (ii) both b_m and b_n belong to the main chain. Table III summarizes our choice of the two anchor blocks and consequent observation period statistics.

[2]The set notations are summarized in Table II for quick reference.

[3]The main chain refers to the longest [36] and heaviest-subtree [43] chain for Bitcoin and Ethereum, respectively.

TABLE III: Observation period statistics

	Preamble block		Concluding block		Population	
	Block #	Date	Block #	Date	# Blks	# Txs
Bitcoin	587,500	2019/07/28	597,500	2019/10/02	10004	21,454,436
Ethereum	8,290,000	2019/08/05	8,620,000	2019/09/25	353,155	36,665,088

C. Measurement Soundness

As public blockchain networks are built on the best-effort serviced overlay network of P2P connections, complete and timed information delivery is not guaranteed. We analyze the completeness of block arrival events (Section III-C1), transaction arrival events (Section III-C2), and their arrival time accuracy (Section III-C3) to underpin the soundness of our measurement.

*1) **Block coverage**:* Blocks can be part of the main chain or they can be stale. We test our observation against the main chain, which is available on a number of block explorer services, to make sure there are no missing blocks.

TABLE IV: Stale blocks (B_{stale}) and transactions (T_{stale})

Region	Bitcoin		Ethereum	
	# Blks	# Txs	# Blks	# Txs
AS-East	1	41,720	23,160	1,170,706
AS-South	3	41,653	23,160	1,194,018
EU-North	4	41,693	23,159	1,192,435
EU-West	2	41,668	23,157	1,123,680
NA-Northeast	3	41,703	23,160	1,212,210
US-Central	1	41,683	23,159	1,212,389
US-East	3	41,692	23,160	1,186,488
US-West	4	41,672	23,155	1,351,847

The main chain blocks are *all observed* while stale blocks are partially observed; stale blocks being partially observed is expected as blockchain nodes are not obliged to propagate stale blocks. Table IV shows the observed number of stale blocks across our measurement points. The stale blocks account for only $\sim$0.04% for Bitcoin while $\sim$7.02% of the observed blocks are stale for Ethereum. The stale block probability of Bitcoin has been decreasing [14], [23] and Ethereum's stale block probability is 5$\sim$10% larger than known uncle rates (calculated only with reported uncles) as our measurement captures unreported stale blocks as well [49].

*2) **Transaction coverage**:* Incomplete transaction measurement causes a subset of transactions which are included in blocks never being observed during measurement. Aggregating transactions observed from different monitors can compensate for this incompleteness. The *unseen* transactions are acquired by subtracting the observed transactions from the transactions included in the main chain blocks.

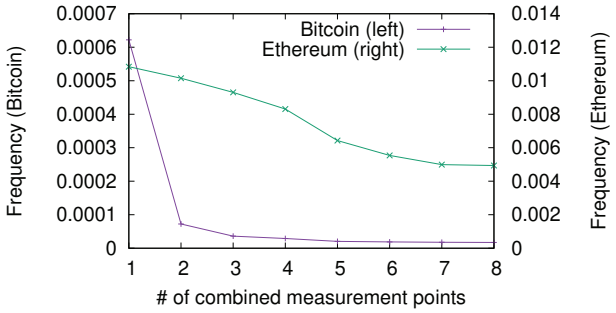


Fig. 3: Unseen transaction probability distribution

Figure 3 shows the probability of unseen transactions (i.e., $n(\mathrm{T}^{M}_{\mathrm{unseen}})/n(\mathrm{T}_{\mathrm{obs}})$) as a function of the number of combined measurement points. As combining all measurement point transaction arrivals results in a converged frequency of $\sim$0.05% for Bitcoin and $\sim$0.5% for Ethereum, it implies that unseen transactions will be still observed even if more monitors are deployed to the network.

TABLE V: Ethereum unseen transactions

From address	Public name	Transaction count	Miner
0xea67⋆	Ethermine	128145 (72.42%)	Ethermine
0x2a65⋆	DwarfPool	22955 (12.97%)	DwarfPool
0x8fd0⋆	unknown	11346 (6.41%)	Spark Pool
0x829b⋆	F2Pool	4453 (2.51%)	F2Pool
0x4c54⋆	Hiveon Pool	1135 (0.64%)	Spark Pool
0x99c8⋆	BeePool	468 (0.26%)	BeePool
0xdba7⋆	unknown	348 (0.20%)	UUPool
0x464b⋆	FKPool	340 (0.19%)	FKPool
0xf687⋆	unknown	217 (0.12%)	AntPool
0xa801⋆	unknown	151 (0.09%)	AntPool
0xbe7c⋆	unknown	126 (0.07%)	AntPool
0xb96f⋆	unknown	101 (0.06%)	AntPool
Sum		169785 (95.87%)	

TABLE VI: Bitcoin unseen transactions

Category	Transaction count	
Zero-fee transaction	148	(50.68%)
Zero-fee transaction dependent transaction	48	(16.44%)
Multi-sig transaction input	43	(14.73%)
Conflict with stale transaction	27	(9.25%)
OP_RETURN transaction output	26	(8.90%)
Sum	292	(100.00%)

Unseen transactions are characterized by the strong evidence that implies *the unseen transactions are indeed never advertised to the network*. Table V categorizes 95.87% of Ethereum unseen transactions by using the *from* addresses of the transactions.[4] Public names are retrieved from Etherscan [17]. A miner is specified in the table only if all blocks that include the unseen transactions with the from address are congruently mined by that single miner. It shows that most of the unseen transactions are indeed the block miner's own transactions.

Table VI shows unseen Bitcoin transactions categorized by the identified reasons. $\sim$67% of the unseen transactions are either zero-fee or dependent transactions to a zero-fee transaction. We define a transaction $t \in \mathsf{transactions}(b)$ is a dependent transaction if at least one of input transactions of t is included in the same block b. Bitcoin Core [12] implementation does not propagate transactions with zero transaction fee. Also, a dependent transaction cannot be mempooled until the transaction to which it is dependent is mempooled; consequently, it cannot be relayed to peers. The Bitcoin Core mempool implementation justifies the inexistence of zero-fee and their dependent transactions from our measurement. Lastly, multi-sig transaction inputs often imply off-chain protocols where signed transactions are exchanged out-of-band [40] and having OP_RETURN transaction output implies special purpose protocols of Bitcoin, e.g., sidechains, where their relationship with miners require further investigation [2].

The presence of unseen transactions, either by private transaction mining or potential out-of-band delivery of transactions, has a subtle security implication. One obvious side-effect is privatization of transaction fees; the transaction fees of privately mined ones are exclusively obtainable by the private miner alone, unless the network is fully utilized. If the network is fully utilized, a miner is forced to include her private transactions at the expense of not including other transactions, as the block capacity is limited.

*3) **Timing Accuracy**:* Consider an ordered arrival timestamp of block b observed at all measurement points M: $\{ts_i \mid 0 \leq i < n(M)\} = \{\mathsf{arrival}(b)^m \mid \forall m \in M\}$ where $ts_i < ts_j$ for all $i < j < n(M)$. We evaluate the *tightness* of our measured timing information with $\Delta(ts_0, ts_1)$ and how far the arrival times can be dispersed among the measurement points with $\Delta(ts_0, ts_7)$ where $\Delta()$ computes the timewise difference. Figure 4 depicts the distribution of the two variables for all block arrival events. It shows that the timestamps are tight, as more than 90% of the blocks arrived with a time difference less than 100ms. Although Bitcoin block arrivals can be delayed by another block time interval ($\sim$ 10 minutes), $\sim$80% of the blocks are not dispersed by more than one second. The long tails – implying that block arrival times can be very prolonged in a small number of cases – are commonly observed for Bitcoin in the literature [14], [15]. Transaction arrival times are much tighter as $\Delta(ts_0, ts_1)$ is less than one second for more than 95% of the transactions for both blockchains.[5]

[4]We use prefixes followed by ⋆ in presenting hash values, e.g., address (i.e., pubkey hash) and *txid*, for brevity. We understand this does not entirely anonymize identity as the readers can verify their results against ours by matching the prefixes.

[5]The distribution is not shown due to space limitation.

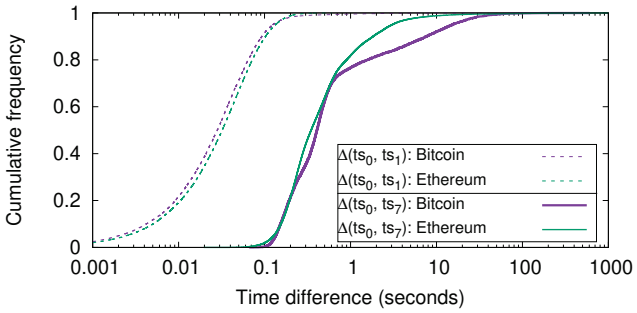
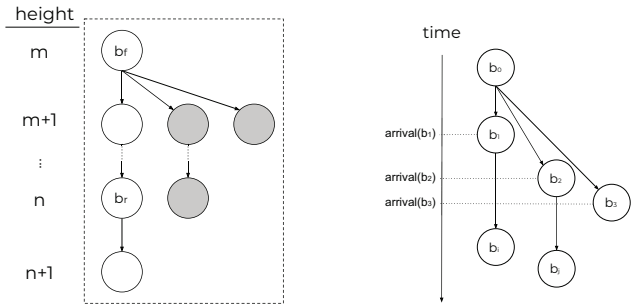


Fig. 4: Block arrival time difference

IV. Measurement Results

A. Blockchain Forks



(a) Fork episode (b) Fork resolve example

Fig. 5: Fork analysis model

A non-overlapping fork episode e at block height $[m, n)$ is defined by identifying two blocks:

- fork initiating block $b_f \in B_{main}$ where $\mathsf{height}(b_f) = m$ and $n(\mathsf{children}(b_f)) > 1$
- fork resolving block $b_r \in B_{main}$ where $\mathsf{height}(b_r) = n$ and $\mathsf{children}(b_i) = \emptyset$ for all $b_i \in S_n$

where $S_n = \{b_i \mid \mathsf{height}(b_i) = n \text{ and } b_i \notin B_{main}\}$. Figure 5(a) illustrates our fork episode model.

We define three characteristic variables; (i) its run length $\mathsf{run\text{-}length}(e) = n - m$, (ii) the weight (i.e., the number of stale blocks in the episode) $\mathsf{weight}(e) = \sum_{h=m}^{n}(n(S_h))$, and (iii) the fork degree (i.e., the maximum number of fork branches) $\mathsf{degree}(e) = \max_{m \le h \le n}(n(S_h) + 1)$. Figure 6 shows how characteristic variables are distributed for Ethereum and our findings are summarized as follows:

- Due to its long block time (i.e., ∼10 minutes) and low stale block probability (i.e., <0.05%), statistical characterization of Bitcoin fork episodes suffers from a lack of sample events. Even three years of live measurement yield less than a hundred fork episodes as we only expect to observe less than 30 fork events per year.
- Ethereum fork episodes mostly run one block height with a single additional branch. The fork degree (i.e., maximum # of branches) is as high as three for ∼5% of the episodes and the forks are not immediately resolved, i.e., having two or more run length, in about 3% of the cases.

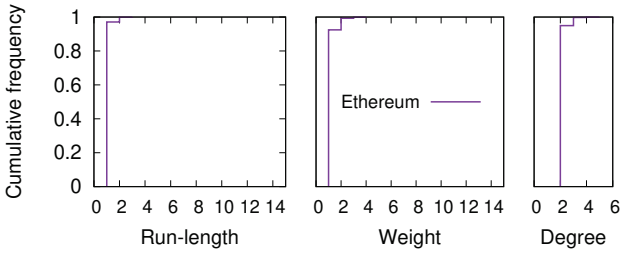
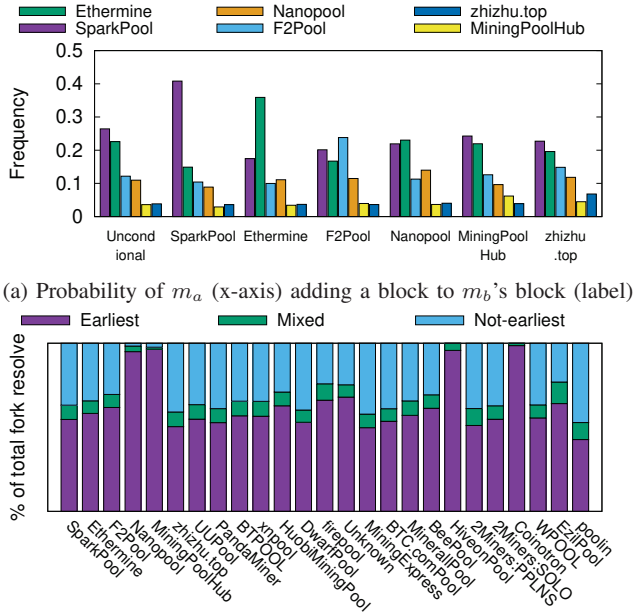


Fig. 6: Ethereum fork episode characteristics



(a) Probability of m_a (x-axis) adding a block to m_b's block (label)

(b) Probability of adding to the earliest propagated block

Fig. 7: Ethereum fork resolve statistics

Our most interesting finding is that Ethereum miners often do not add blocks to the earliest advertised block during fork episodes. Consider an illustrative fork episode depicted in Figure 5(b). The episode begins as the three blocks (i.e., b_1, b_2, and b_3) are added to the fork initiating block b_0. The immediately following blocks (i.e., b_i and b_j) will have their previous blockhash identifying one of the three previous blocks of the miner's choice. The block b_i, mined by some miner, may be added to the earliest block (i.e., b_1); the other block b_j is mined by a different miner who chooses a different block (i.e., b_2) to add to. To evaluate miners' preference of blocks to which succeeding blocks are added in the event of blockchain forks, we define the conditional probability of miner m_a adding a block to a previous block mined by m_b (by not adding to any of a

different miner's block) in fork episodes for all m_a and m_b:

$$\Pr(\mathsf{miner}(b_i) = m_b \mid \mathsf{miner}(b_j) = m_a \text{ and } \mathsf{prev}(b_j) = b_i \text{ and } \exists b_k : \mathsf{height}(b_k) = \mathsf{height}(b_i))$$

Figure 7(a) shows a histogram of the above probability. It is apparent that the top ranked miners significantly prefer adding blocks to their own mined blocks over other miners' blocks. Figure 7(b) depicts the same probability by labeling whether or not the chosen block (by m_a) is the earliest propagated one. The 'earliest' label indicates that the block, to which the newly mined block (by m_a) is added, has been observed as the earliest one to arrive congruently among all eight measurement locations.[6] For example, the top ranked pool *SparkPool* chose the earliest arrived block ∼55% of the time, 3409 out of 6234 blocks; among the other 2825 blocks that were added to non-earliest previous blocks, *SparkPool*'s own blocks were chosen over others 1211 times, while other pools' blocks were preferred to *SparkPool*'s only 24 times. In the meantime, *Nanopool* chose the earliest one a majority of the time. *For the top 25 ranked mining pools, only four of them added blocks to the earliest arrived blocks, more than 90% of the time.* This contradicts the earliest propagated block endorsement assumption that is broadly adopted in mining related research [13], [19], [28] [7].

One possible explanation of the phenomenon is that the mining pools adopt their own block preferring strategy as a simple precautionary defense against yet unknown selfish mining pools; the profitability of selfish mining largely depends on the probability of honest pools adding to the attacking pool's branch. We further discuss implications in Section V.

B. Block Time Gaps

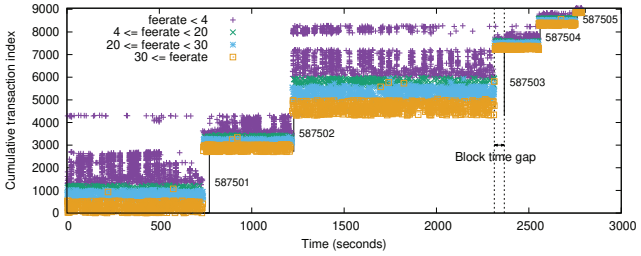


Fig. 8: Bitcoin transaction and block arrival illustration. Points are depicted at the time of transaction arrivals which are included in the block whose arrival is depicted with a vertical bar. Transactions are broken down to four feerate groups.

One common observation we have made for both PoW blockchains is that there are non-negligible time gaps between the block arrival time and the arrival time of the latest transaction that is included in said block. Figure 8 illustrates an example of a 50-second-long time gap observed in Bitcoin block #587503.

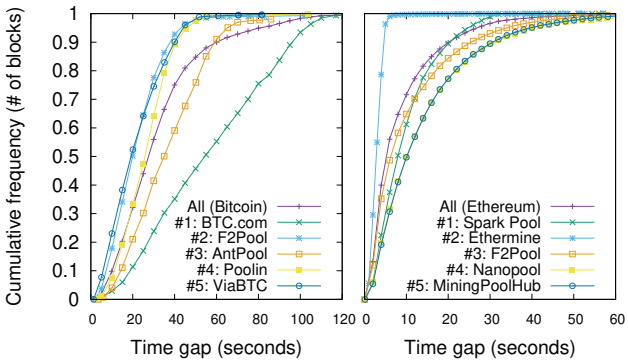


Fig. 9: Time gap dist. of Bitcoin (left) and Ethereum (right)

Figure 9 shows the distribution of time gaps for the Bitcoin and Ethereum blocks we have measured. The median gaps are around 25 seconds and 5 seconds long for Bitcoin and Ethereum, respectively. The time gap distribution diverges depending on the miner; the blocks mined by the top ranked Bitcoin mining pool *BTC.com* show much larger time gaps, e.g., a median time gap of 60 seconds. As for Ethereum mining pools, the blocks mined by No. 2 mining pool *Ethermine* exhibit much smaller time gaps.

The prevalence of large time gaps can be explained by either (or both) of two hypotheses:

1) Mining pools deliberately delay publication of their mined blocks by privately working on (or sharing only with colluding miners) the mined-but-not-published block to improve their chances of finding the next block earlier with an elevated risk of other competing miners finding a valid block during the delay in the meantime.
2) Mining pools (or their workers) do not update the block proposal frequently, so that newly-arrived-but-higher-fee transactions are not promptly included in blocks.

To examine the first theory, the probability distributions of miner m_b mining a block, conditional to the previous block's miner being m_a for all m_a and m_b are examined. Figure 10 shows the probability of Bitcoin blocks mined by m_b (label), conditional to the previous block being mined by m_a (x-axis) for top-ranked miners. As the probability distribution is not different across the miners (m_a), we speculate that the theory is not convincing.[8]

As for the second theory, mining pools construct their own block proposals by choosing a set of transactions from their mempool (or transaction pool in the case of Ethereum) to be included in their currently mining block. As the capacity of a block is limited, it is rational for miners to maximize their

[6]As block arrival times are measured in eight different locations, the arrival order can be conflicting. The 'mixed' label represents the conflicting cases.

[7]We do not claim that the behavior is selfish or against consensus rules. The mining pool protocols (e.g., *Stratum* [33]) merely define a way of exchanging mining related messages while we believe that the mining pools equip with customized algorithms to meet their user policy and operational goals.

[8]We cannot disprove the theory due to ignorance of block proposal time. One may think the timestamp value in blocks should reflect the block proposal time, but it is informed that the timestamps are not accurate due to the *Ntime rolling* feature of the popular mining protocol *Stratum* [6], [46].

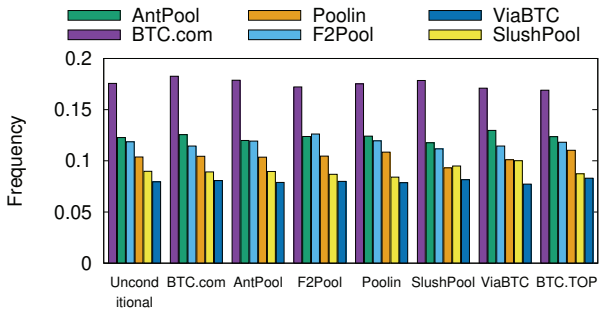


Fig. 10: Conditional Bitcoin block mining probability dist.

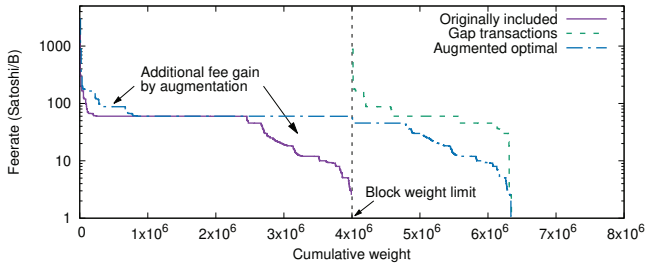


Fig. 11: Example weight-feerate diagram of the original and augmented optimal block proposals

profit by choosing the set with the highest total transaction fee. In consequence, having an up-to-date block proposal (i.e., maximizing the transaction fee by including late-arrived-but-higher-fee transactions in the proposal) appears to be a preferred strategy of PoW mining, as the update gives the same probability of solving the puzzle:

$$\Pr(\mathcal{H}(nonce \mid \hat{b}_o) \leq \mathrm{T}(d)) = \Pr(\mathcal{H}(nonce \mid \hat{b}_u) \leq \mathrm{T}(d))$$

for an original and updated block proposal $\hat{b}_o \neq \hat{b}_u$, respectively, where $\mathcal{H}(\cdot)$ is a cryptographic hash function and $\mathrm{T}(d)$ is a target function with difficulty parameter d.

However, today's popular mining pool protocols support features letting mining workers expand their nonce search space to extra block header fields, such as the time field [46], to maximize the number of trials achievable with hardware accelerated hash computation with a single block proposal fetch from the pool.[9] Given the observation that infrequent block proposal update is the norm, then the question is, *what is the cost of sluggish block proposal update?*

The optimization problem of transaction scheduling [45] can be modeled as a *0-1 knapsack problem* that is NP-hard:

$$\textit{maximize} \sum_{i=1}^{n} v_i x_i$$

$$\textit{subject to} \sum_{i=1}^{n} w_i x_i \leq C \textit{ and } x_i \in \{0,1\}$$

where v_i and w_i are the transaction fee and weight of the i-th Bitcoin transaction, respectively. As for Ethereum, w_i is set to the actual *gas* spent by the i-th transaction, retrieved from the transaction receipts; v_i is the product of *gas price* and w_i. The block capacity C is 4,000,000 for Bitcoin and the *gas limits* of the original block proposal for Ethereum blocks are set individually. The optimal solutions for an updated block proposal $\hat{b}_u$ by augmenting the original block proposal $\hat{b}_o$ with the transactions that arrived during the time gap are computed.

Figure 11 shows the weight-feerate diagram (as in [45]) of an example Bitcoin block. The line labeled 'originally included' shows the original diagram of the block. The line labeled 'gap transactions' shows the weight-feerate diagram of the transactions that arrived during the time gap of the block. Finally, 'augmented optimal' shows the optimal solution of the sum. The area between the two lines in the range $[0{:}4{\times}10^6]$ represents the transaction fee additionally obtainable by including the gap transactions.

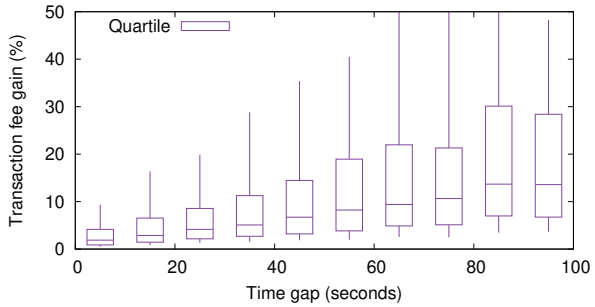


Fig. 12: Bitcoin transaction fee gain as a function of time gap

Figure 12 shows the calculated Bitcoin transaction fee gain – 10th, 25th, 50th, 75th, and 90th percentile – as a function of the time gap. As expected, the transaction fee gain is an increasing function of the time gap. Overall, the transaction fee increases 5.15% on average from 0.2476 to 0.2604 BTC per block for Bitcoin. The Ethereum transaction fee gain is 40.39% on average with an increase in its per-block transaction fee from 0.0649 to 0.0911 ETH. *To summarize, the mining pools of both PoW blockchains give up a non-negligible portion of transaction fees due to infrequent block proposal update. It is also notable that the time gaps, and consequent transaction fee losses, vary by mining pool.*

Conversely, mining pools might not be motivated to optimize their transaction scheduling, as the block reward is currently much larger than the transaction fee reward. The mining reward is the sum of the block reward and transaction fee reward. Considering the block rewards of 12.5 BTC for Bitcoin and 2 ETH for Ethereum, the overall block reward gains by optimization are only 0.1% and 1.3%, respectively.

C. Stale Transactions

Stale transactions are network-transmitted transactions that are not included in blocks. Table IV shows the stale transaction counts observed at each location. The stale transactions

[9]Today's ASICs search over the entire 32-bit nonce field in 100ms [33].

account for 0.20% and 3.69% of Bitcoin and Ethereum transactions, respectively; Ethereum transactions had an order of magnitude more chance of being stale than the other.

Stale transactions are mostly attributable to *conflict* for both blockchains. Table VII summarizes the identified reasons of transactions being stale. A stale Bitcoin transaction is attributable to *conflict* if there are one or more transactions sharing at least one transaction input (i.e., *vin*). An Ethereum transaction is attributable to *conflict* if there is another transaction with the same *nonce* from the same account. A widely known reason for two or more transactions being in conflict is transaction replacement such as replace-by-fee [47]. On the other hand, slightly more than 5% of Bitcoin transactions are stale because at least one of their transaction inputs are also stale.

On the other hand, roughly one out of ten stale Ethereum transactions are attributable to improper ordering of *nonce*: *nonce* is too high (9.83%) or too low (0.02%). Last but not least, 7.27% of stale transactions are not included in blocks because the account is not sufficiently funded.[10] An Ethereum account can be insufficiently funded when executing a transaction for a number of reasons. We find these stale transactions are particularly interesting and study them further in the following section.

TABLE VII: Identified reasons of stale transactions

Reason	Transaction count	
	Bitcoin	Ethereum
Conflict	39,716 (94.51%)	1,208,588(89.24%)
Stale transaction input	2,307 (5.49%)	-
Nonce (too high)	-	133,161 (9.83%)
Nonce (too low)	-	251 (0.02%)
Insufficient fund	-	9,847 (7.27%)
Sum	42023(100.00%)	1,351,847(99.82%)

D. Transaction Ambiguity

We have identified cases where transactions can be stale, i.e., not included in any blocks, since the account is not sufficiently funded (see Table VII). While there are only around 10,000 such transactions (uniquely identified with their *txids*) during our measurement period, they are repeatedly advertised, so that a single monitor node has received around 20 million such transaction arrival events; a single transaction of this kind arrives at a node around 2,000 times on average. The most frequently occuring transaction arrived at our monitor node more than 5,000 times over almost two months. The question is, *why do such transactions keep hitting the node by floating around the network for such a long time?*

We find that this phenomenon is unique to the blockchains adopting the account balance model. In the account model, the state in which a transaction should be executed is *ambiguous*. The UTXO model, on the contrary, mandates to specify the source of fund by listing the executed transactions and their index (i.e., *vout*); in consequence, the token that a transaction spends is *not ambiguous*. While the unambiguous UTXO balance model requires a valid transaction to designate *unspent* transaction outputs, the account balance model presumes the account balance checking is required in validating a newly informed transaction; failing to have sufficient funds to execute a transaction results in an *insufficiently funded transaction*.

[10] An Ethereum account should have $balance \geq value + gas \times gas\ price$ to execute the transaction [48].

TABLE VIII: Ethereum transaction validation process. Validation steps are performed in order.

Step	Invalidation reason
0	Transaction hash is already known
1	Transaction size exceeds pre-defined limit
2	*value* is negative
3	*gas* exceeds block gas limit
4	Transaction signature is not valid
5	*gasPrice* is under pre-defined limit
6	*nonce* is misordered (i.e., too low)
7	$value + gas \times gasPrice$ exceeds account balance
8	*gas* is under minimum transaction gas

A consequence of transaction ambiguity is that a transaction, which spends funds belonging to an account, can be composed even before the funding transaction is informed to the transactor. To be technically precise, consider two transactions:

- **Funding transaction t_{fund}** transferring some amount of funds from the address `0xface⋆` to `0xfeed⋆`
- **Withdrawal transaction $t_{withdraw}$** transferring `0xfeed⋆`'s token funded via t_{fund} to the address `0xbadd⋆`

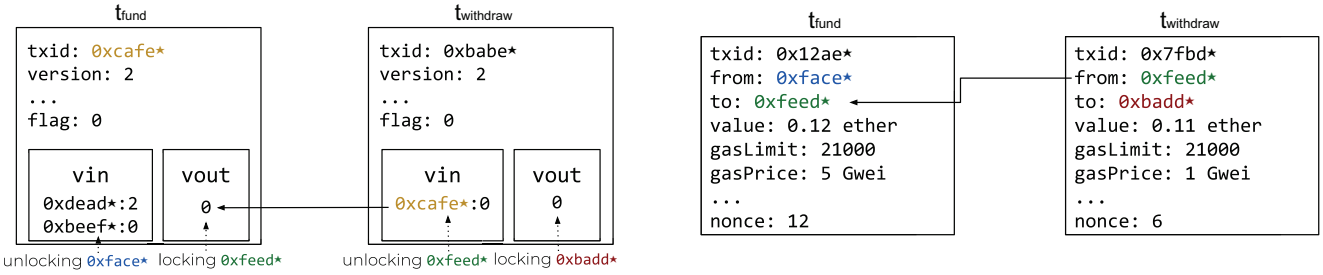
Figure 13 illustrates the composition of t_{fund} and $t_{withdraw}$ for the two different balance models:

- **UTXO model** - the knowledge of t_{fund}'s *txid* (i.e., `0xcafe⋆`) is required to compose $t_{withdraw}$. To fill up *vin* of $t_{withdraw}$, `0xcafe⋆` should be informed – via transaction advertisement or a published block including t_{fund} – to the composer of $t_{withdraw}$, while the transaction id `0xcafe⋆` cannot be determined without the knowledge of a valid signature generated with the secret key corresponding to the public key hash `0xface⋆`.
- **Account model** - a valid $t_{withdraw}$ can be composed by signing with the secret key of `0xfeed⋆` even before the account is funded by t_{fund}. Composition of $t_{withdraw}$ does not require the knowledge of t_{fund}'s *txid*.

A scenario[11] exploiting transaction ambiguity is depicted in Figure 14. Suppose that the attacker (`0xbadd⋆`) has acquired the secret key of the victim (`0xfeed⋆`).[12] Starting from the state where the victim's account (`0xfeed⋆`) is not funded, the victim is expecting some deposit from a third person (`0xface⋆`). The victim wants to transfer the prospective funds to her counterpart (`0xacee⋆`) while the attacker wants to steal it by transferring the funds to her account (`0xbadd⋆`) beforehand. The attacker composes a withdrawal (to his

[11] Similar cases are reported in the community [37], [38].

[12] Private keys can be stolen by malware [10], key misplacement [35], or computations exploiting software vulnerabilities [18].



(a) Bitcoin (UTXO model)

(b) Ethereum (Account model)

Fig. 13: Funding and corresponding withdrawal transaction composition example

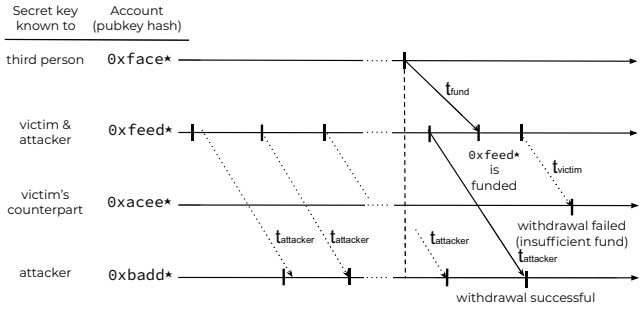


Fig. 14: Transaction ambiguity exploit scenario

account 0xbadd⋆) transaction $t_{attacker}$, exploiting transaction ambiguity, and propagates $t_{attacker}$ to the network so that $t_{attacker}$ gets included in the same (or immediately following) block, which includes t_{fund}. The victim initiates her own withdrawal transaction t_{victim} without success, after the block is published.

TABLE IX: Ethereum transaction statistics

Category	Count	TPS
Transaction arrival events	891,335,182	200.73
Transaction hashes	37,140,702	8.36
Included (unstale) transaction hashes	35,407,892	7.97
Insufficiently funded transaction hashes	9,955	0.002
Insufficiently funded transaction arrival events	20,699,439	4.66

Table IX summarizes the number of transactions and their arrival events that a single monitor node has received during our observation period. Redundant arrivals of normal transactions , that are queued in *txpool* and are finally included in blocks, do not go through the transaction validation process, as they are filtered by their *txid* at a preliminary step, i.e., step 0 of Table VIII. The already known (by *txid*) transactions do not invoke the validation process. On the other hand, every redundant arrival of insufficiently funded transactions – which cannot be pooled, as they are invalidated at the step 7 – triggers the validation process, only to fail at the step 7. Consequently, insufficiently funded transactions incur more than half of the transaction validation overhead (4.66 TPS) compared to that of normal transactions (7.97 TPS), even though they are few (0.002 TPS).

Propagation of insufficiently funded transactions is implementation dependent. Even though there are several Ethereum client implementations out there, Geth [16] and Parity [39] account for 95% of the node population [27]; three out of four nodes run Geth, i.e., the official Ethereum client implementation. Geth does not relay transactions which fail the validation process; however, Parity P2P implementation relays some of them.[13] Figure 15 shows that the peers running Parity advertise duplicate insufficiently funded transactions while Geth nodes do not relay duplicate ones. Even though Geth nodes do not relay invalid transactions, an attacker can DoS a node with continuous transaction advertisement.

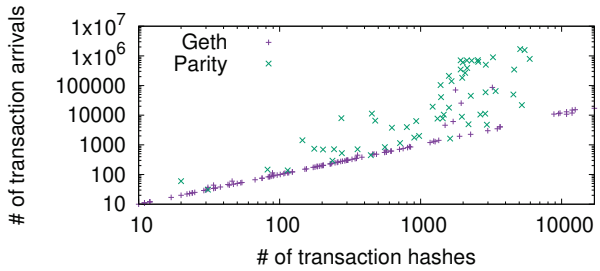


Fig. 15: The number of transaction hashes versus the number of transaction arrivals for Geth and Parity Ethereum peers

V. Discussion

In this section, we discuss security implication on two blockchain attacks: partitioning and mining attacks.

*1) **Partitioning Attacks**:* Partitioning attacks exploit vulnerable peer management mechanisms of Bitcoin [26] and Ethereum [31]. Such attacks are also examined in the context of ISP's BGP hijacking [1]. The attacks often rely on implementation vulnerabilities for efficient eclipsing of peers while known vulnerabilities are getting addressed; for example, the lack of parallel block download sessions causing temporary denial of blocks (reported in [24] and fixed by recommending three concurrent download sessions in [41]) let attackers successfully partition the node without eclipsing all

[13]We discovered a bug (relaying insufficiently funded transactions) in Parity light client implementation. We have responsibly disclosed this issue to OpenEthereum to receive a bounty payouts of $2,000 USD.

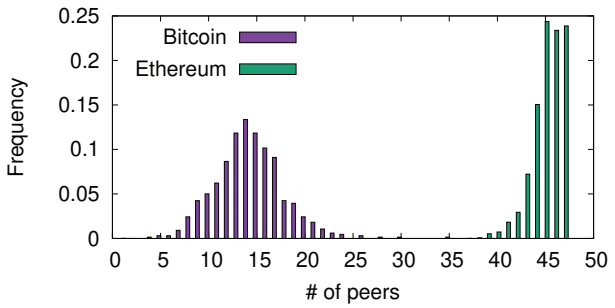


Fig. 16: Peer redundancy of block advertisement

peer connections [1]. A simple countermeasure of increasing the number of peer connections, on the other hand, is applied to the reference public blockchain implementations [11], [21], to inflate attack complexity.

We investigate the degree of a peer's advertisement redundancy on newly generated blocks and transactions to evaluate the efficacy of partitioning attacks and the countermeasure of increasing the number of peer connections. Figure 16 shows how redundantly newly generated block information is offered by peers. Bitcoin nodes have maintained ∼43 peer connections on average and Ethereum nodes record all 50 peer connections established during the entire observation period.[14] For each newly generated block, one third of peers advertised the block header to our Bitcoin node and more than 90% of Ethereum peers signaled availability of the blocks. In consequence, successfully partitioning a node requires compromise of peers proportional to the total number of peer connections. Increasing the number of peer connections will burden the attacker linearly as expected.

In the meantime, we emphasize DoS attack vectors such as the one exploiting transaction ambiguity (Section IV-D) become more fatal with an abundance of peer connections; defending partitioning attacks by increasing peer connections not only inflates communication overhead but also amplifies the impact of prospective DoS attacks.

*2) **Mining Attacks**:* Cryptocurrency mining has been a popular subject in assessing the security of Bitcoin; selfish mining [19], block withholding attack [13], and a mixed strategy [28] have been studied. Although it is shown that selfish behavior in mining is always more profitable than being honest, without need for a mutual agreement of not attacking each other [28], we find selfish mining is not the preferred strategy. All selfish mining strategies will result in blockchain forks, but we observe a very low fork probability (i.e., 0.04%) for Bitcoin. On the other hand, selfish mining on Ethereum has been studied only recently to unanimously claim that selfish mining is also profitable for Ethereum miners; however, it is less profitable than Bitcoin due to its uncle reward system [20]. Our result in Section IV-A shows that a majority of Ethereum miners choose to add to their own produced blocks over the earliest propagated blocks. How does this affect the selfish mining strategy?

The selfish mining strategy assumes a parameter γ which defines the fraction of honest miners' mines on the attacker's fork branch.[15] If the attacking pool has a network-wise superior position in block propagation (i.e., $\gamma > 0.5$), the selfish mining strategy is profitable for pools with proportional mining power $\alpha > 25\%$ [19]. Absolute network inferiority (i.e., $\gamma = 0$ due to a defensive mining strategy such as the one presented in Section IV-A) increases the required mining power to 33%; the 33% mining power appears to be significantly harder to achieve in the real world [22].

As for the FAW (Fork-After-Withholding) attack strategy [28], the network capability parameter C defines the probability that an attacker's FPoW (Full Proof-of-Work) block (during infiltration mining) is selected as a main chain block. C affects both the attack reward and whether the miner's dilemma holds or not in the mutual attacking game. As an example case (Section 9 of [28]) of a two-pool game between F2Pool and BitFury, the absolute network inferiority drops C from 0.914 to 0.3 to make it a dilemmatic game again. Unfortunately, miners' own block preferences cannot be directly transferred to a single parameter of γ or C; it calls for a model change from previous mining studies.

VI. Related Work

Historical transaction data of Bitcoin have often been studied for economic aspects such as cryptocurrency abuse [29], anti-money laundering [34], and monetary flow of businesses [30]. However, historical blockchain data, usually available from block explorer services, are not sufficient for our research purpose. We have surveyed the popular block explorer services of Bitcoin [4], [5], [7] and Ethereum [5], [8], [17] to see the availability of information – beyond what are recorded in the main chain block data – we have utilized in our study, namely:

- **Stale blocks/transactions** - no Bitcoin block explorers provide stale block information. All Ethereum block explorers provide only reported stale blocks (i.e., uncle blocks [48]), while our blockchain fork analysis (Section IV-A) cannot be performed without the knowledge of *unreported* stale blocks. No block explorers provide stale transaction information.
- **Block/transaction arrival times** - all block explorers provide block timestamps that are recorded in the block headers, not the actual times of arrival. Our blockchain fork and gap analysis require measured block arrival times at multiple locations. All block explorers show transaction timestamps identical to the block timestamp to which the transactions belong.[16] Our block time gap analysis (Section IV-B) and transaction analysis (Section IV-D) require precise transaction arrival times.

Researchers have performed blockchain measurement studies to assess the degree of decentralization [22], to investigate

[14] See Table I for the default peer management configuration.

[15] The parameter comes into play in case (f) of [19].

[16] Recent Bitcoin transactions have different timestamps than the block timestamps in [4] while the timestamps are only at a granularity of one minute.

demographic information of network nodes [15], [27], to study mining pool behavior [44], and to evaluate security of PoW blockchains [23] against known attacks such as selfish mining [19]. Our work expands the understanding of previous security evaluations with the key findings acquired from our large-scale transaction and block measurement. As for the blockchain and cryptocurrency attacks, we have summarized the existing attacks in Section V for better readability.

VII. Conclusion

We have measured transaction and block arrival events of the two public blockchains to evaluate underlying mechanisms in security perspectives. We have shared our key findings and their implications. Finally, we have reported a subtle but critical vulnerability of Ethereum that can be exploited by an attacker to launch a network-wide DoS attack.

Acknowledgment

The authors would like to thank Seungwon Woo, Daegeun Yoon, and Changhyun Lee for helping us with the measurement instrumentation. This work was supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government [21ZR1300, Research on Intelligent Cyber Security and Trust Infra].

References

[1] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: Routing attacks on cryptocurrencies," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 375–392.
[2] M. Bartoletti and L. Pompianu, "An analysis of bitcoin op_return metadata," in *International Conference on Financial Cryptography and Data Security*. Springer, 2017, pp. 218–230.
[3] bitcoincash.org. [Online]. Available: https://www.bitcoincash.org
[4] Blockchain.com. [Online]. Available: https://www.blockchain.com
[5] Blockchair. [Online]. Available: https://blockchair.com
[6] R. Bowden, H. P. Keeler, A. E. Krzesinski, and P. G. Taylor, "Block arrivals in the bitcoin blockchain," *arXiv:1801.07447*, 2018.
[7] BTC.com. Bitcoin block explorer. [Online]. Available: https://btc.com
[8] ——. Ethereum block explorer. [Online]. Available: https://eth.btc.com
[9] V. Buterin. (2013) Ethereum white paper. [Online]. Available: https://ethereum.org/en/whitepaper/
[10] C. Cimpanu. (2020) Chrome extension caught stealing crypto-wallet private keys. [Online]. Available: https://www.zdnet.com/article/chrome-extension-caught-stealing-crypto-wallet-private-keys/
[11] B. Core. (2016) Bitcoin core version 0.19.0.1 release note. [Online]. Available: https://bitcoin.org/en/release/v0.19.0.1
[12] ——. (2020). [Online]. Available: https://github.com/bitcoin/bitcoin
[13] N. T. Courtois and L. Bahack, "On subversive miner strategies and block withholding attack in bitcoin digital currency," *arXiv preprint arXiv:1402.1718*, 2014.
[14] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *IEEE P2P 2013 Proceedings*. IEEE, 2013, pp. 1–10.
[15] J. A. D. Donet, C. Pérez-Sola, and J. Herrera-Joancomartí, "The bitcoin p2p network," in *International Conference on Financial Cryptography and Data Security*. Springer, 2014, pp. 87–102.
[16] G. Ethereum. (2020) Go ethereum. [Online]. Available: https://github.com/ethereum/go-ethereum
[17] Etherscan. Ethereum explorer. [Online]. Available: https://etherscan.io
[18] I. S. Evaluators. (2019) Ethercombing: Finding secrets in popular places. [Online]. Available: https://www.ise.io/casestudies/ethercombing/
[19] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *International conference on financial cryptography and data security*. Springer, 2014, pp. 436–454.
[20] C. Feng and J. Niu, "Selfish mining in ethereum," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 1306–1316.
[21] E. foundation. (2019) Geth v1.9.0. [Online]. Available: https://blog.ethereum.org/2019/07/10/geth-v1-9-0/
[22] A. E. Gencer, S. Basu, I. Eyal, R. Van Renesse, and E. G. Sirer, "Decentralization in bitcoin and ethereum networks," in *International Conference on Financial Cryptography and Data Security*. Springer, 2018, pp. 439–457.
[23] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 3–16.
[24] A. Gervais, H. Ritzdorf, G. O. Karame, and S. Capkun, "Tampering with the delivery of blocks and transactions in bitcoin," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 692–705.
[25] Google. Google cloud. [Online]. Available: https://cloud.google.com
[26] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network," in *24th USENIX Security Symposium*, 2015, pp. 129–144.
[27] S. K. Kim, Z. Ma, S. Murali, J. Mason, A. Miller, and M. Bailey, "Measuring ethereum network peers," in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 91–104.
[28] Y. Kwon, D. Kim, Y. Son, E. Vasserman, and Y. Kim, "Be selfish and avoid dilemmas: Fork after withholding (faw) attacks on bitcoin," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 195–209.
[29] S. Lee, C. Yoon, H. Kang, Y. Kim, Y. Kim, D. Han, S. Son, and S. Shin, "Cybercriminal minds: an investigative study of cryptocurrency abuses in the dark web," in *Network and Distributed System Security Symposium*. Internet Society, 2019, pp. 1–15.
[30] M. Lischke and B. Fabian, "Analyzing the bitcoin network: The first four years," *Future Internet*, vol. 8, no. 1, p. 7, 2016.
[31] Y. Marcus, E. Heilman, and S. Goldberg, "Low-resource eclipse attacks on ethereum's peer-to-peer network." *IACR Cryptol. ePrint Arch.*, vol. 2018, p. 236, 2018.
[32] Monero. (2020). [Online]. Available: https://web.getmonero.org
[33] P. Moravec and J. Čapek. (2020) Stratum v2 — the next generation mining pool protocol. [Online]. Available: https://stratumprotocol.org
[34] M. Möser, R. Böhme, and D. Breuker, "An inquiry into money laundering tools in the bitcoin ecosystem," in *2013 APWG eCrime researchers summit*. Ieee, 2013, pp. 1–14.
[35] M. Munford. (2019) How i lost £25,000 when my cryptocurrency was stolen. [Online]. Available: https://www.bbc.com/news/business-49177705
[36] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019.
[37] nickoneill. (2018) A christmas mystery: sweepers and zero gas price transactions. [Online]. Available: https://www.reddit.com/r/ethereum/comments/7lx1do/a_christmas_mystery_sweepers_and_zero_gas_price/
[38] ——. (2018) Ethereum sweeper bots gone wild. [Online]. Available: https://www.reddit.com/r/ethereum/comments/839f9o/ethereum_sweeper_bots_gone_wild/
[39] Parity. (2020) Parity ethereum client - open ethereum. [Online]. Available: https://www.parity.io/ethereum/
[40] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016.
[41] B. I. Proposal. (2016) Bip 0152. [Online]. Available: https://en.bitcoin.it/wiki/BIP_0152
[42] Ripple. (2013-2020) Xrp. [Online]. Available: https://ripple.com/xrp/
[43] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 507–527.
[44] C. Wang, X. Chu, and Q. Yang, "Measurement and analysis of the bitcoin networks: A view from mining pools," *arXiv preprint arXiv:1902.07549*, 2019.
[45] B. Wiki. (2014) Miner fees. [Online]. Available: https://en.bitcoin.it/wiki/Miner_fees
[46] ——. (2014) Stratum mining protocol. [Online]. Available: https://en.bitcoin.it/wiki/Stratum_mining_protocol
[47] ——. (2018) Replace by fee. [Online]. Available: https://en.bitcoin.it/wiki/Replace_by_fee
[48] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, 2014.
[49] Ycharts. (2020) Ethereum uncle rate. [Online]. Available: https://ycharts.com/indicators/ethereum_uncle_rate