Blockchain Double Spending With Low Mining Power and Network Delays

CHRIS NATOLI, University of Sydney
PARINYA EKPARINYA, University of Sydney
GUILLAUME JOURJON, CSIRO
VINCENT GRAMOLI, EPFL, University of Sydney and Redbelly Network

Traditional blockchain systems offer a secure way of tracking the ownership of digital assets as long as the attacker does not control a large portion of the overall computational or mining power. They typically require participants to generate a proof-of-work before proposing a block at a given index of the chain. To choose one block among the candidate blocks at the same index, Nakamoto's consensus, Ghost and the original Ethereum's consensus select, respectively, the longest branch, the heaviest subtree and the branch with the most difficult crypto-puzzles. This allows an attacker who can generate proofs-of-work faster than others to double spend by overwriting any given branch.

In this paper, we present a double spending attack, called the Balance attack, that simply needs to delay some messages. This result sheds new lights on an important, often implicit, assumption of the blockchain, *synchrony*, under which the transmission delay of any message should be within a known upper bound. We show that the attack succeeds with high probability on the protocols of the two largest blockchain systems in market capitalization, Bitcoin and Ethereum. To quantify the impact of our attack, we replicated the blockchain network run by fifty financial institutions and achieved double spending in less than 20 minutes. Finally, we demonstrate the success of the attack empirically by modifying the `geth` software and hijacking BGP in a controlled distributed system whose distribution of mining power is set to the distribution observed on the Ethereum main blockchain.

1 INTRODUCTION

Blockchain systems are distributed implementations of a chain of blocks. Each node can issue a cryptographically signed transaction to transfer digital assets to another node or can create a new block of transactions, and append this block to its current view of the chain. Due to the distributed nature of this task, multiple nodes may append distinct blocks at the same index of the chain before learning about the presence of other blocks, hence leading to a forked chain or a *tree*. For nodes to eventually agree on a unique state of the system, they apply a common strategy that selects a unique branch of blocks in this tree.

Two of the most popular blockchains systems, Bitcoin [52] and Ethereum [77], choose the longest and heaviest branch, respectively. If the attacker owns an important part of the total computational power or *mining power* of the system to grow a local branch of the blockchain faster than the rest of the system, then this attacker can eventually impose its own branch to all participants. Except in the ZLB blockchain [62], controlling a majority of resources is typically sufficient for the attack to succeed. The participants will have no choice but to accept this particular branch, hence aborting the conflicting transactions they previously committed in other branches. When the attacker benefits from this situation to "re-spend" the same coins he spent in one of these aborted transactions, we call the result of this attack a *double spending* [64]. Thankfully, the highly distributed nature of the blockchain makes it unlikely for a single node to own a sufficiently large part of the mining power of the system.

While the consensus of Bitcoin is reached by selecting the longest branch among multiple ones, the consensus of Ethereum, originally inspired by the Ghost protocol [66], is to select the heaviest branch in terms of the expected mining power to generate it. The basic version of Ghost differs slightly from both the selection strategy of Ethereum (v1.x) and The Merge and starts from the first block, also called the *genesis block*, and iteratively selects the root of the heaviest subtree to construct the common branch. Even if nodes create many blocks at the same index of the blockchain, their mining power is not wasted but counted in the selection strategy. In particular, the number of these "sibling"

blocks increase the chance that their common ancestor block is selected in favor of another candidate block mined by the attacker. Ethereum Merge combines proof-of-stake with a consensus algorithm closer to the original idea of Ghost.[1]

In this paper, we demonstrate that for the three strategies to reach consensus (longest branch, heaviest branch and heaviest subtree) an attacker can compensate a low mining power by delaying selected messages to double spend. To this end, we propose a simple attack, called the *Balance attack* [54]: an attacker transiently disrupts communications between subgroups of similar mining power. During this time, the attacker issues transactions in one subgroup, say the *transaction subgroup*, and mines blocks in another subgroup, say the *block subgroup*, up to the point where the tree of the block subgroup outweighs, with high probability, the tree of the transaction subgroup. This strategy allows the attacker to mine a light and small branch possibly in isolation of the rest of the network before merging its branch to one of the competing branches in order to influence the branch selection process.

We analyse the feasibility of this attack theoretically on an Ethereum network in similar settings as R3 [61], a consortium of world-wide financial institutions. As opposed to a fully private network scenario, such a consortium network involves different institutions, possibly competitors, that may not trust each other. At the time of our experiment, R3 deployed Ethereum [63] on 50 machines owned by distinct financial institutions to test banking services. Since then, R3 has experimented with various blockchains and developed the Corda distributed ledger [16], that can run in a trusted network but cannot cope with the arbitrary behavior of a single node.[2] In this R3 Ethereum setting, we demonstrate that a single institution can steal assets of other institutions if some messages are delayed during less than 20 minutes. This theoretical analysis is interesting to generalize to other settings: it shows that if the difficulty of the crypto-puzzle in the proof-of-work drops or if the attacker owns a third of the computational power, then even a few second delay in delivering messages becomes sufficient for an attacker to double spend. Note that Ethereum also offers proof-of-authority as an alternative to proof-of-work, however, it had recently been proved vulnerable [29]. Finally, for the sake of ethical responsibility, we communicated, prior to publication, all identified vulnerabilities to the R3 management and technical teams who shared their experimental settings with us.

To evaluate empirically the feasibility of the Balance Attack, we ran it on a distributed network of machines with a mining power distribution close to the one of the Ethereum main chain [28]. Bitcoin and Ethereum are popular for their main chain that is available to internet users without specific permissions. Many researchers already studied the impact of network delays on Bitcoin [25, 37, 56, 59, 66] while a few have considered Ethereum [53]. Some attacks consist of delaying the propagation of blocks [34, 73], others require to delay the messages between one node and the rest of the network [42, 53] or completely partition the network [11, 53], however, none of these attacks are full fledged: they do not give the double spending risks resulting from attacking the network. To address this limitation, we experiment the risks for an attacker to double spend when executing the balance attack that includes hijacking the routing protocol used by the machines running Ethereum. To this end, we gathered mining power information of the Ethereum public blockchain and deployed the Ethereum protocol on a sandboxed network of distributed machines with BGP routers configured with OpenStack. We configure the machines to mimic the distribution of mining power of Ethereum by restricting the CPU quantum of each machine with linux `cgroups`. We then hijacked BGP to redirect the traffic towards the attacker for some time during which we ran a Balance Attack and demonstrated that the attack of the largest miner would succeed 8 times out of 10. It is interesting that our attack cannot be as easily applied to the Ethereum main chain.

[1] https://blog.ethereum.org/2020/02/12/validated-staking-on-eth2-2-two-ghosts-in-a-trench-coat/#ghosts-and-their-opinions-on-forks.
[2] We discussed with the R3 development team about a Byzantine fault tolerant prototype version that they recommended us to avoid due to lack of stability.

In particular, we finally collected topological information of the Ethereum main chain to conclude that the lack of information on the location of mining pool makes it more difficult to apply our BGP-hijacking attack to the Ethereum main chain than to an instance of Ethereum within an Internet-based consortium (cf. Section 7.4).

Our Balance attack permitted to identify an important assumption of Ethereum that had, to our knowledge, not been stated explicitly before [54]. This *synchrony* assumption common in the distributed computing literature indicates that every message should be delivered in an amount of time lower than a known upper bound. Synchrony is often used to simplify the problem and some researchers have already modelled precisely the Bitcoin network by assuming synchrony [35]. While strong guarantees can clearly be proved under this assumption, the guarantees to expect if this assumption is violated remained unclear. The problem is that mainstream blockchains, whether they are fully public or involve a consortium of institutions, rely on large networks, like the Internet, in which one cannot predict the delay of messages. The Balance attack reveals that delaying messages can be dramatic for blockchains, leading to double-spending even with a small portion of the mining power and raises interesting challenges in the design of safer blockchains [24, 68] that we discuss at the end of this paper. As a result of releasing the Balance Attack, the inventor of Ethereum acknowledged publicly that the security of Ethereum requires this synchrony assumption to hold [69], an assumption that remains absent of the numerous successive versions of the Ethereum white and yellow papers that appeared over the past years [17, 21, 76, 78].

In Section 2, we give preliminary definitions and states the problem. In Section 3, we present the algorithm to run the attack. In Section 4, we analyze Nakamoto's consensus, Ghost and Ethereum's consensus. In Section 5, we simulate a Balance attack in the context of the R3 consortium network. In Section 6, we run the Balance attack in a private Ethereum testnet. In Section 7, we hijack BGP to demonstrate the feasibility of the attack on an emulation of the Ethereum main chain. In Section 8, we propose solutions that cope with the Balance Attack by avoiding forks. Section 9 presents the related work and Section 10 concludes.

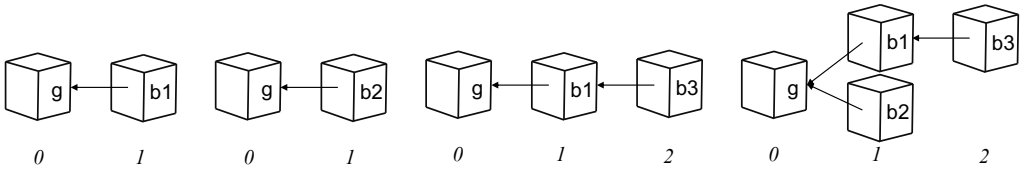
2 PRELIMINARIES

In this section we model a simple distributed system that implements a blockchain abstraction as a directed acyclic graph. We specify at a high-level Nakamoto's consensus, Ghost and Ethereum's consensus protocols and explain when the prefix of the blockchain is considered persistent in that it can no longer be mutated.

2.1 A simple distributed model for blockchains

We consider a communication graph $G = \langle V, E \rangle$ with nodes V connected to each other through fixed communication links E. Nodes are part of a blockchain system $S \in \{\mathsf{bitcoin}, \mathsf{ethereum}\}$ and can act as clients by issuing transactions to the system and/or servers by *mining*, the action of trying to combine transactions into a block. For the sake of simplicity, we consider that each node possesses a single account and that a *transaction* issued by node p_i is a transfer of digital assets or *coins* from the account of the source node p_i to the account of a destination node $p_j \neq p_i$. Each transaction is uniquely identified and broadcast to all nodes in a best-effort manner. We assume that a node re-issuing the same transfer multiple times creates as many distinct transactions.

Nodes that mine are called *miners*. We refer to the computational power of a miner as its *mining power* and we denote the total mining power t as the sum of the mining powers of all miners in V. Each miner tries to group a set T of transactions it heard about into a block $b \supseteq T$ as long as transactions of T do not conflict and that the account balances remain non-negative. For the sake of simplicity in the presentation, the graph G is static meaning that no nodes can join and leave the system, however, nodes may fail as described in Section 2.1.2. As we consider a network potentially



(a) view ℓ_1 (b) view ℓ_2 (c) view ℓ_3 (d) view ℓ_3

Fig. 1. The global state ℓ_0 of a blockchain results from the union of the distributed local views ℓ_1, ℓ_2 and ℓ_3 (maintained by correct nodes) of the blockchain

shared by other applications and subject to contention, we assume that the system is *partially synchronous* in that there is a bound on the delay of messages but that this bound can be large and is not known by the algorithm [27].

2.1.1 Miners try to create new blocks. Miners have the role of creating blocks, by provably solving a hashcash crypto-puzzle [15], a puzzle that consists of finding a cryptographic hash that starts with a given number of zeros. Given a global threshold and the block of largest index the miner knows, trying to solve a crypto-puzzle consists of repeatedly selecting a nonce and applying a pseudo-random function to this block and the selected nonce until a result lower than the threshold is obtained. Upon success the miner creates a block that contains the successful nonce as a proof-of-work as well as the hash of the previous block, hence fixing the index of the block, and broadcasts the block. As there is no known strategy to solve the crypto-puzzle, the miners simply keep testing whether randomly chosen numbers solve the crypto-puzzle. The mining power is thus expressed in the number of hashes the miner can test per second, or H/s for short. The *difficulty* of this crypto-puzzle, defined by the threshold, limits the rate at which new blocks can be generated by the network.

2.1.2 The failure model. We assume the presence of an *attacker* that can control malicious nodes that together own a relatively small fraction $0 < \rho < \frac{1}{2}$ of the total mining power of the system. The nodes controlled by the attacker may not follow the protocol specification in an arbitrary way [46], however, we assume a public key infrastructure to sign transactions so that they cannot impersonate other nodes while issuing transactions. A node that is not malicious is *correct*.

2.1.3 The forkable blockchain abstraction. Let the *blockchain* be a directed acyclic graph (DAG) $\ell = \langle B, P \rangle$ such that blocks of B point to each other with pointers P (pointers are recorded in a block as a hash of the previous block) and a special block $g \in B$, called the *genesis block*, does not point to any block but serves as the common first block.

Algorithm 1 Blockchain construction at node p_i

1: $\ell_i = \langle B_i, P_i \rangle$, the local blockchain at node p_i is a directed acyclic
2: graph of blocks B_i and pointers P_i

3: receive-blocks$(\langle B_j, P_j \rangle)_i$: ▷ upon reception of blocks
4: $B_i \leftarrow B_i \cup B_j$ ▷ update vertices of blockchain
5: $P_i \leftarrow P_i \cup P_j$ ▷ update edges of blockchain

Algorithm 1 describes the progressive construction of a forkable blockchain at a particular node p_i upon reception of blocks from other nodes by simply aggregating the newly received blocks to the known blocks (lines 3–5). As every added block contains a hash to a previous block that eventually leads back to the genesis block, each block is associated with a fixed index. By convention we consider the genesis block at index 0, and the blocks at j hops away from the

genesis block as the blocks at index j. As an example, consider the simple blockchain $\ell_1 = \langle B_1, P_1 \rangle$ depicted in Figure 1(a) where $B_1 = \{g, b_1\}$ and $P_1 = \{\langle b_1, g \rangle\}$. The genesis block g has index 0 and the block b_1 has index 1.

2.1.4 Forks as disagreements on the blocks at a given index. As depicted by views ℓ_1, ℓ_2 and ℓ_3 in Figures 1(a), 1(b) and 1(c), respectively, correct nodes may have a different views of the current state of the blockchain. Note that we ignore the views that malicious nodes might have of the blockchain since malicious nodes can forge their own view of any kind.

In particular, it is possible for two (correct) miners p_1 and p_2 to mine almost simultaneously two different blocks, say b_1 and b_2. If neither block b_1 nor b_2 was propagated early enough to nodes p_2 and p_1, respectively, then both blocks would point to the same previous block g as depicted in Figures 1(a) and 1(b). Because network delays are not predictable, a third (correct) node p_3 may receive the block b_1 and mine a new block without hearing about b_2. The three correct nodes p_1, p_2 and p_3 thus end up having three different local views of the same blockchain, denoted $\ell_1 = \langle B_1, P_1 \rangle$, $\ell_2 = \langle B_2, P_2 \rangle$ and $\ell_3 = \langle B_3, P_3 \rangle$.

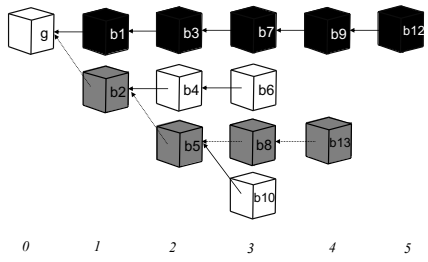


Fig. 2. Nakamoto's consensus selects the main branch as the longest branch (in black) whereas the Ghost consensus protocol follows the heaviest subtree (in grey).

We refer to the *global blockchain* as the directed acyclic graph $\ell_0 = \langle B_0, P_0 \rangle$ representing the union of these local blockchain views, denoted by $\ell_1 \cup \ell_2 \cup \ell_3$ for short, as depicted in Figure 1, and more formally defined as follows:

$$\begin{cases} B_0 & = \cup_{\forall i} B_i, \\ P_0 & = \cup_{\forall i} P_i. \end{cases}$$

The point where distinct blocks of the global blockchain DAG have the same predecessor block is called a *fork*. As an example Figure 1(d) depicts a fork with two branches pointing to the same block, g in this example.

In the remainder of this paper, we refer to the DAG as a *tree* rooted in g with upward pointers, where children blocks point to their parent block.

2.1.5 Main branch selection. To resolve the forks and define a deterministic state agreed upon by all nodes, a blockchain system must select a *main branch*, as a unique sequence of blocks, based on the tree. Building upon the generic construction (Alg. 1), we present three consensus protocols: Nakamoto's consensus protocol (Alg. 2) present in Bitcoin [52], the Ethereum's consensus protocol (Alg. 3) [77] and the Ghost protocol (Alg. 4) [66] that inspired Ethereum's strategy.

Nakamoto's consensus algorithm. The difficulty of the crypto-puzzles used in Bitcoin produces a block every 10 minutes in expectation. The advantage of this long period, is that it is relatively rare for the blockchain to fork because blocks are rarely mined during the time others are propagated to the rest of the nodes.

Algorithm 2 depicts the Bitcoin-specific pseudocode that includes Nakamoto's consensus protocol to decide on a particular block at index i (lines 8–19) and the choice of parameter m (line 6) explained later in Section 2.2. When a fork occurs, Nakamoto's protocol resolves it by selecting the longest branch as the main branch (lines 8–16) by iteratively selecting the root of the deepest subtree (line 11). When process p_i is done with this pruning, the resulting branch becomes the main branch $\langle B_i, P_i \rangle$ as observed by the local process p_i. Note that the pseudocode for checking whether a

Algorithm 2 Nakamoto's consensus protocol at node p_i

```
6:  m, the number of blocks to be appended after the block containing
7:      tx, for tx to be committed in Bitcoin

8:  get-main-branch()_i:                                   ▷ select the longest branch
9:      b ← genesis-block(B_i)                             ▷ start from the blockchain root
10:     while b.next ≠ ⊥ do                                ▷ prune shortest branches
11:         block ← argmax_{c∈children(b)} {depth(c)}      ▷ root of deepest subtree
12:         B ← B ∪ {block}                                ▷ update vertices of main branch
13:         P ← P ∪ {⟨block, b⟩}                           ▷ update edges of main branch
14:         b ← block                                      ▷ move to next block
15:     end while
16:     return ⟨B, P⟩                                      ▷ returning the Bitcoin main branch

17: depth(b)_i:                                            ▷ depth of tree rooted in b
18:     if children(b) = ∅ then return 1                   ▷ stop at leaves
19:     else return 1 + max_{c∈children(b)} depth(c)       ▷ recurse at children
```

Algorithm 3 Ethereum's consensus protocol at node p_i

```
1:  m, the number of blocks to be appended after the block containing
2:      tx, for tx to be committed in Ethereum

3:  get-main-branch()_i:                                   ▷ select the branch with highest total difficulty
4:      b ← genesis-block(B_i)                             ▷ start from the blockchain root
5:      while b.next ≠ ⊥ do                                ▷ prune shortest branches
6:          block ← argmax_{c∈children(b)} {difficulty(c)} ▷ root of the subtree with highest accumulated difficulty
7:          B ← B ∪ {block}                                ▷ update vertices of main branch
8:          P ← P ∪ {⟨block, b⟩}                           ▷ update edges of main branch
9:          b ← block                                      ▷ move to next block
10:     end while
11:     return ⟨B, P⟩                                      ▷ returning the Ethereum main branch

12: difficulty(b)_i:                                       ▷ total difficulty of tree rooted in b
13:     if children(b) = ∅ then return b.difficulty        ▷ stop at leaves
14:     else return b.difficulty + max_{c∈children(b)} difficulty(c)   ▷ recurse at children
```

block is decided and a transaction committed based on this parameter m is common to Bitcoin and Ethereum, it is thus deferred to Alg. 5.

2.1.6 Ethereum's consensus algorithm. Ethereum adjusts the difficulty of crypto-puzzles in every new block based on the gap time between the latest block in the main branch and its predecessor to maintain a block time of 12-15 seconds. In comparison to the 10-minute block time in Bitcoin, the shorter block time in Ethereum allows the blockchain to accept and commit transactions more frequently. Algorithm 3 depicts, in pseudocode, how Ethereum's consensus protocol selects a particular block at index i (lines 3–14) in the versions 1.x of Ethereum. Again, the choice of the parameter m (line 6) is deferred to Section 2.2. Ethereum's protocol resolves a fork by selecting a branch with the highest total difficulty as the main branch (lines 3–11) by iteratively selecting the root of the subtree with highest accumulated difficulty of crypto-puzzles (line 6). Although not shown here for the sake of clarity, when the difficulties are the same, then the algorithm selects the shortest branch; but if the lengths are also the same, then some implementation (like go ethereum) tosses a coin to select either branch with the same probability. Note that this algorithm results from a careful check of the Ethereum source code as we could not find this information in any documentation.

2.1.7 The Ghost consensus algorithm. Ghost is a branch selection algorithm that has always inspired the Ethereum consensus algorithm, however, Ethereum (v1.x) has not used it in favor of the most difficult branch consensus presented

Table 1. Parameter m based on the suggestion from Bitcoin Project and Ethereum founder's posts.

System	m	Description
Bitcoin	0	Recommended by Bitcoin Project as "Mostly reliable" [60]
Bitcoin	1	Recommended by Bitcoin Project as "Highly reliable" [60]
Bitcoin	5	Recommended by Bitcoin Project as a minimum for high-value bitcoin transfers [60]
Ethereum	9	Mentioned by Vitalik in his blog post for a 17-second blockchain [18]
Ethereum	11	Suggested value by Vitalik in one of his posts [19]
Bitcoin	29	Recommended by Bitcoin Project during emergencies to allow human intervention [60]

in Section 2.1.6. As Ethereum generates new valid blocks more frequently in expectation than Bitcoin, Ethereum improves the throughput (transactions per second) but also favors transient forks as miners are more likely to propose new blocks without having heard about the latest mined block(s) yet. To avoid wasting large mining efforts while resolving forks, the Ghost (Greedy Heaviest Observed Subtree) consensus protocol depicted in Alg. 4 accounts for the, so called *uncles*, blocks of discarded branches. In contrast to the previously presented consensus protocols, the Ghost protocol iteratively selects, as the successor block, the root of the subtree that contains the largest number of blocks (line 11 and lines 17–19). Ethereum Merge is closer to a variant of the Ghost protocol [20].

Algorithm 4 The Ghost consensus protocol at node p_i

```
 6: m, the number of blocks to be appended after the block containing
 7:    tx, for tx to be committed (Ethereum as an example)

 8: get-main-branch()_i:                                  ▷ select the branch with the most nodes
 9:    b ← genesis-block(B_i)                             ▷ start from the blockchain root
10:    while b.next ≠ ⊥ do                                ▷ prune lightest branches
11:       block ← argmax_{c∈children(b)} {num-desc(c)}    ▷ root of heaviest tree
12:       B ← B ∪ {block}                                 ▷ update vertices of main branch
13:       P ← P ∪ {⟨block, b⟩}                            ▷ update edges of main branch
14:       b ← block                                       ▷ move to next block
15:    end while
16:    return ⟨B, P⟩.                                     ▷ returning the Ethereum main branch

17: num-desc(b)_i:                                       ▷ number of nodes in tree rooted in b
18:    if children(b) = ∅ then return 1                   ▷ stop at leaves
19:    else return 1 + ∑_{c∈children(b)} num-desc(c)      ▷ recurse at children
```

The main difference between Nakamoto's and Ethereum's consensus protocols, and Ghost is depicted in Figure 2. The black blocks represent the main branch selected by Nakamoto's consensus protocol and Ethereum's consensus protocol provided that the difficulty of all solved crypto-puzzle is identical. The grey blocks represent the main branch selected by Ghost.

2.2 Decided blocks and committed transactions

A blockchain system S must define when the block at an index is agreed upon. To this end, it has to define a point in its execution where a prefix of the main branch can be "reasonably" considered as persistent.

More precisely, there must exist a parameter m provided by S for an application to consider a block as *decided* and its transactions as *committed*. This parameter is typically $m_{bitcoin} = 5$ in Bitcoin and $m_{ethereum} = 11$ in Ethereum as the suggested safety margins depicted in Table 1. Note that these two choices do not lead to the same probability of success [36].

Definition 1 (Transaction commit). *Let $\ell_i = \langle B_i, P_i \rangle$ be the blockchain view at node p_i in system S. A transaction tx is* committed *if there exists a node p_i such that tx is* locally committed. *For a transaction tx to be* locally committed *at p_i, the conjunction of the following properties must hold in p_i's view ℓ_i:*

(1) *Transaction tx has to be in a block $b_0 \in B_i$ of the main branch of system S. Formally, $tx \in b_0 \wedge b_0 \in B'_i : \langle B'_i, P'_i \rangle =$ get-main-branch$()_i$.*
(2) *There should be a subsequence of m blocks $b_1, ..., b_m$ appended after block b. Formally, $\exists b_1, ..., b_m \in B_i$: $\langle b_1, b_0 \rangle, \langle b_2, b_1 \rangle, ..., \langle b_m, b_{m-1} \rangle \in P_i$. (In short, we say that b_0 is* decided.*)*

Property (1) is needed because nodes eventually agree on the main branch that defines the current state of accounts in the system; blocks that are not part of the main branch are ignored. Property (2) is necessary to guarantee that the blocks and transactions currently in the main branch will persist and remain in the main branch. Before these additional blocks are created, nodes may not have reached consensus regarding the unique blocks b at index j in the chain. This is illustrated by the fork of Figure 1 where nodes consider, respectively, the pointer $\langle b_1, g \rangle$ and the pointer $\langle b_2, g \rangle$ in their local blockchain view. By waiting for m blocks where m is given by the blockchain system, the system guarantees with a reasonably high probability that nodes will agree on the same block b. Note that the property is not prefix-closed in the sense that get-main-branch may return a chain that is not necessarily a prefix of another branch returned later.

Algorithm 5 Checking transaction commit at node p_i

```
20: is-committed(tx)_i:                                  ▷ check whether transaction is committed
21:     ⟨B'_i, P'_i⟩ ← get-main-branch()                 ▷ pick main branch Alg. 2, 3 or 4
22:     if ∃b_0 ∈ B'_i : tx ∈ b_0 ∧ ∃b_1, ..., b_m ∈ B_i :   ▷ tx in main branch
23:             ⟨b_1, b_0⟩, ⟨b_2, b_1⟩..., ⟨b_m, b_{m-1}⟩ ∈ P_i then   ▷ enough blocks
24:         return true
25:     else return false
```

For example, consider a fictive blockchain system with $m_{fictive} = 2$ that selects the heaviest branch (Alg. 4, lines 8–16) as its main branch. If the blockchain state was the one depicted in Figure 2, then blocks b_2 and b_5 would be decided and all their transactions would be committed. This is because they are both part of the main branch and they are followed by at least 2 blocks, b_8 and b_{13}. (Note that we omit the genesis block as it is always considered decided but does not include any requested transaction.)

3 THE BALANCE ATTACK

In this section, we present the Balance attack, a novel form of attack that affects proof-of-work blockchains, especially Bitcoin and Ethereum. Its novelty lies in identifying subgroups of miners of equivalent mining power and delaying messages between them rather than entering a race to mine blocks faster than others.

The Balance attack demonstrates a fundamental limitation of main proof-of-work systems in that they are not immutable because an attacker can rewrite the blockchain by delaying messages, hence allowing it to double spend. We thus say that these blockchain systems are *corruptible* as defined below.

Definition 2 (Corruptibility). *A blockchain system is* corruptible *if an attacker can:*

(1) *make the recipient of a transaction tx observe that tx is committed and*
(2) *later remove the transaction tx from the main branch.*

with probability $1 - \varepsilon$, where $0 < \varepsilon < 1$ is defined by the attacker.

Note that the attacker selects ε close to 0 to maximize the chances of the attack based on the delay of messages as we will illustrate in line 13 of Algorithm 6. In Section 4, we will show that, as this delay increases, ε decreases exponentially fast towards 0, and the probability of attack success thus increases exponentially fast towards 1.

The Balance attack is simple: while the attacker disrupts communications between correct subgroups of equivalent mining power, it simply issues transactions in one subgroup. The attacker then mines sufficiently many blocks in another subgroup to ensure with high probability that the subtree of another subgroup outweighs the transaction subgroup's. Even though the transactions are committed, the attacker can rewrite with high probability the blocks that contain these transactions by outweighing the subtree containing this transaction.

The attacker in the Balance attack disrupts communications between subgroups of miners to delay messages, particularly block propagations. The attacker may employ one or more network attack techniques to break TCP/IP connectivity among miners and partition them into subgroups for a period of time. The broken connectivity of transport or network layer prevents message delivery on application layer, which in turn delays block propagations between subgroups until the network partition ends. As the attacker does not need to know or modify payload of network traffic to delay messages, the attack applies to both unencrypted and encrypted communication.

One could benefit from delaying messages only between the merchant and the rest of the network by applying the Eclipse attack [42] to Ethereum. The Eclipse attack was designed against Bitcoin and eclipsing one node of Bitcoin is non trivial: it requires to restart the node's protocol in order to control all the logical neighbors the node will eventually try to connect to. While a Bitcoin node typically connects to few logical neighbors, an Ethereum node typically connects to tens of nodes, making the problem harder. Another option would be to isolate a subgroup of smaller mining power than another subgroup, however, it would make the attack only possible if the recipients of the transactions are located in the subgroup of smaller mining power. Although possible this would limit the generality of the attack, because the attacker would be constrained on the transactions it can override.

Note that the Balance attack inherently violates the persistence of the main branch prefix and is enough for the attacker to double spend. The attacker has simply to identify the subgroup that contains merchants and create transactions to buy goods from these merchants. After that, it can issue the transactions to this subgroup while propagating its mined blocks to at least one of the other subgroups. Once the merchant shipped goods, the attacker stops delaying messages. Based on the high probability that the tree seen by the merchant is outweighed by another subtree, the attacker could reissue another transaction transferring the exact same coin again.

3.1 Attacker Model

As in the Dolev-Yao model [26], we assume the attacker, who has the control over the Byzantine participants, can intercept messages, delay, or delete them. However, we assume the attacker cannot modify messages as it does not have enough power to forge signatures. We assume the adversary has access to, or controls, network infrastructure that performs routing for the blockchain network. We discuss in more details this threat model feasibility in Section 3.4.

3.2 Executing a Balance Attack

For the sake of simplicity, let us fix $k = 2$ so that we consider two subgraphs of miners (the proof generalizes to the case $k > 2$ by partitioning more subgraphs). We consider subgraphs $G_1 = \langle V_1, E_1 \rangle$ and $G_2 = \langle V_2, E_2 \rangle$ of the communication graph $G = \langle V, E \rangle$ so that each subgraph has half of the mining power of the system. Let $E_0 = E \setminus (E_1 \cup E_2)$ be the set of edges that connects nodes of V_1 to nodes of V_2 in the original graph G. Let τ be the communication delay introduced by

the attacker on the edges of E_0. The lower bound on τ is selected with sufficient probability, later explained in the proof of Theorem 7.

As indicated in Algorithm 6, the attacker can introduce a sufficiently long delay τ during which the miners of G_1 mine in isolation of the miners of G_2 (line 13). As a consequence, different transactions get committed in different series of blocks on the two blockchains locally viewed by the subgraphs G_1 and G_2. Let b_2 be a block present only in the blockchain viewed by G_2 but absent from the blockchain viewed by G_1. In the meantime, the attacker issues transactions spending coins C in G_1 (line 14) and mines a blockchain starting from the block b_2 (line 16). Before the delay expires the attacker sends his blockchain to G_2. After the delay expires, the two local views of the blockchain are exchanged. Once the heaviest branch to which the attacker contributed is adopted, the attacker can simply reuse the coins C in new transactions (line 19).

Algorithm 6 The Balance attack initiated by attacker p_i

1: **State:**
2: $G = \langle V, E \rangle$, the communication graph
3: pow, a mapping from of a node in V to its mining power in $\mathbb{R}$
4: $\ell_i = \langle B_i, P_i \rangle$, the local blockchain at node p_i is a directed acyclic
5: graph of blocks B_i and pointers P_i
6: $\rho \in (0; 1)$, the portion of the mining power of the system owned by
7: the attacker p_i, with $0 < \rho < 0.5$
8: d, the difficulty of the crypto-puzzle currently used by the system

9: balance-attack$(\langle V, E \rangle)_i$: ▷ starts the attack
10: Select $k \geq 2$ subgraphs $G_1 = \langle V_1, E_1 \rangle, ..., G_k = \langle V_k, E_k \rangle$:
11: $\sum_{\forall v \in V_1} \text{pow}(v) \approx ... \approx \sum_{\forall v' \in V_k} \text{pow}(v')$
12: Let $E_0 = E \setminus \cup_{\forall 0 < i \leq k} E_i$ ▷ attack communication channels
13: Stop communications on E_0 during $\tau \geq \frac{(1-\rho)6d\log(\frac{4}{\varepsilon})}{\rho^2 t}$ seconds
14: Issue transaction tx crediting a merchant in graph G_i with coins C
15: Let b_2 be a block appearing in G_j but not in G_i
16: Start mining on ℓ_i immediately after b_2 ▷ contributed to correct chain
17: Send blockchain view ℓ_i to some subgraph G_j where $j \neq i$
18: When τ seconds have elapsed, stop delaying communications on E_0
19: Issue transaction tx' that double-spends coins C ▷ double spending is successful

3.3 Exploiting the knowledge about the network

As indicated by the state of Algorithm 6, an attacker has to be knowledgeable about the current settings of the blockchain system to execute a Balance attack. In fact, the attacker must have information regarding the logical or physical communication graph, the mining power of the miners or pools of miners and the current difficulty of the crypto-puzzle.

Most of this information about Ethereum is public and can be found online [30]. In particular, the difficulty of the crypto-puzzle, the propagation delay, block uncles and the times between blocks are available. Also, the mining power, the Ethereum client and version, and the latency of any node that chooses to register voluntarily, is automatically made available.

In Section 7.4, we will explain however why finding the necessary connectivity information about miners to attack mainstream blockchains can be more difficult than between a consortium of partners connected through Internet. The interesting aspects of the Balance attack is that it can apply either to the logical overlay or to the physical overlay of the blockchain system. While there exist tools to retrieve the logical overlay topology, like AddressProbe [50], to find

the overlay information of Bitcoin, it can be easier for an attacker of the blockchain system to run a man-in-the-middle attack [28] rather than a BGP hijacking as we demonstrate in Section 7.2.

3.4 Delaying networking communications

While disrupting the communication between subgroups of a blockchain system may look difficult, there have been multiple attacks successfully delaying blockchains messages in the past.

In 2014, a BGP hijacker exploited access to an ISP to steal $83,000 worth of bitcoins by positioning itself between Bitcoin pools and their miners [47]. Some known attacks against Bitcoin involved partitioning the communication graph at the network overlay [10] and at the application overlay [42]. At the network level, a study indicated techniques for autonomous systems to intercept a large amount of bitcoins and evaluated the impact of these network attacks on the Bitcoin protocol [10]. At the application level, some work showed that an attacker controlling 32 IP addresses can "eclipse" a Bitcoin node with 85% probability [42]. Although a number of mitigations have been implemented making network-level eclipse attacks sufficiently difficult [42], there are a number of new techniques that highlight the possibility of new weaknesses that can be used in combination to perform this attack [71, 72]. Similarly, a number of eclipsing techniques have been proposed for the Ethereum blockchain [43, 49] with highlights that show some mitigations cannot be patched due to compatibility concerns.

Starting in September 2016, Ethereum experienced a denial-of-service attack that forced miners to spend a long time accessing memory while executing smart contracts. While it did not lead to double-spending as far as we know it slowed down the entire network [75]. More generally, there are a variety of network attacks [28], known as man-in-the-middle attacks and spoofing attacks, that can be exploited to lead to similar results by relaying the traffic between two nodes through the attacker.

4 ANALYSIS OF THE BALANCE ATTACK

In this section, we show that the Balance attack makes any blockchain system based on Nakamoto's consensus, Ethereum consensus or the Ghost algorithm corruptible. In particular, this translates into double spending with a small portion of the mining power by delaying the network.

4.1 Additional Assumptions

For the sake of simplicity in the proof, we assume that the number of partitions, or groups of isolated correct nodes, is $k = 2$ and $\sum_{\forall v \in V_1} \text{pow}(v) = \sum_{\forall v' \in V_2} \text{pow}(v')$ so that the communication is delayed between only two communication subgraphs G_1 and G_2 of equal mining power. We also assume that the difficulty of all solved cryptopuzzles for the same index is identical in Ethereum[3] and that the block propagation delay of Bitcoin within each partition is null, so that miners always mine on the same branch in the same partition. This is motivated by the fact that the time to propagate blocks is low compared to the time between block creations in Bitcoin as we explained in Section 2.1.5. As we explain below, the same attack analysis applies to the branch selection metrics of all three consensus algorithms. We summarize previous and new notations in Table 2.

4.2 Selection Metrics

[3]Note that this difficulty differs significantly within two subgraphs only if one subgraph has a significantly different mining power than the other.

Before we can analyze the probability of success of the Balance Attack we explain why the analysis is identical for the three proof-of-work blockchain protocols, Bitcoin, Ghost and Ethereum.

Table 2. Notations of the analysis

t	total mining power of the system (in million hashes per second, MH/s)
d	difficulty of the crypto-puzzle (in million hashes, MH)
ρ	fraction of the mining power owned by the malicious miner (in percent, %)
τ	disruption time of communication between subgraphs (in seconds, s)
μ_c	mean of the number of blocks mined by each subgraph during time τ
μ_m	mean of the number of blocks mined by the attacker during time τ
Δ	the maximum difference of mined blocks for the two subgraphs

Let us recall from Section 2.1.5 that Nakamoto's consensus (Alg. 2), the Ethereum consensus (Alg. 3) and Ghost (Alg. 4) measure the depth, the num-desc and the difficulty of a branch, respectively, and select the branch among all candidate branches that maximizes this metric. We thus call the metrics depth, num-desc and difficulty the *selection metrics* of Bitcoin, Ghost and Ethereum, respectively.

The goal is to charaterize the selection metric used in each of proof-of-work blockchain protocols depth, num-desc and difficulty used to select the right branch in Nakamoto's consensus (Alg. 2), Ethereum consensus (Alg. 3) and Ghost (Alg. 4), respectively. As in all these algorithms there is no better strategy for solving the crypto-puzzles than random trials, we consider that each group of miners $G_i, i \in \{1, 2\}$ mines blocks during time τ by performing a series of n trials. It follows that the selection metric contributed by correct miners follows a binomial distribution as indicated by Lemma 3.

Lemma 3. *For each of the studied proof-of-work consensus protocols (Algorithms 2, 3 and 4), the selection metric contributed by each group of connected correct miners during time τ is a random variable that follows a binomial distribution with mean $\mu_c = \frac{(1-\rho)t\tau}{2d}$.*

Proof. By examination of the pseudocode of Nakamoto's consensus (Alg. 2), Ethereum consensus (Alg. 3) and Ghost (Alg. 4), we can see that the three considered selection metrics, depth, num-desc and difficulty, applied to the subtree generated by graphs G_1 and G_2 all follow a similar strategy:

(1) Nakamoto: the num-desc increases by 1 with probability $p = \frac{1}{d}$ as new blocks get appended to one of the descendants of the blockchain prefix common to both subgroups.
(2) Ethereum: as we assume that all blocks have the same difficulty of the crypto-puzzle in Ethereum, it follows that the difficulty of the branch increases by a positive constant, normalized to $c = 1$, with probability $p = \frac{1}{d}$.
(3) Ghost: as we assume that the block propagation delay of Nakamoto's consensus is null, it follows that the depth of the branch increases by 1 with probability $p = \frac{1}{d}$.

For any of the three considered algorithms, we can conclude that to mine blocks, each group of miners among G_1 and G_2 performs a series of $n = \frac{(1-\rho)t\tau}{2}$ independent and identically distributed Bernoulli trials whose outcome is:

$$\begin{cases} 1 & \text{with probability } p = \frac{1}{d}, \\ 0 & \text{with probability } 1 - p. \end{cases}$$

Let the sum of these outcomes for subgraphs G_1 and G_2 be the random variables X_1 and X_2, respectively. The random variables X_1 and X_2 follow a binomial distribution with mean:

$$\mu_c = np = \frac{(1-\rho)t\tau}{2d}. \tag{1}$$

The result follows. □

As the malicious miners own ρ of the total mining power t, a corollary of Lemma 3 implies that a group of malicious nodes cannot contribute more than μ_m as follows.

Corollary 4. *For each algorithm, the selection metric contributed by the malicious miners during time τ cannot exceed $\mu_m = \frac{\rho t \tau}{d}$.*

4.3 Analysis of the Balance Attack

The Balance attack relies on the attacker selecting the branch that will obtain a higher score according to the specific blockchain protocols using selection metrics depth, num-desc and total-diff as described in Nakamoto's consensus, Ghost and Ethereum consensus, respectively. Thus, we first lower-bound the probability $\Pr[\mu_m > \Delta]$ that the contributed score μ_m mined by the attacker is greater than the difference $\Delta = |X_1 - X_2|$ in blocks mined by the two subgraphs G_1 and G_2.

Let us first recall the Bernoulli's inequality.

Fact 5 (Bernoulli's inequality). *$1 + nt \leq (1+t)^n$ for $n \geq 1$ and $t \geq -1$.*

We can now lower-bound the probability $\Pr[\mu_m > \Delta]$ as follows.

Lemma 6. *If the attacker owns a portion $0 < \rho < \frac{1}{2}$ of the total mining power, then at time τ:*

$$\Pr[\mu_m > \Delta] > \left(1 - 2e^{-\frac{\rho^2}{3(1-\rho)^2}\mu_c}\right)^2. \tag{2}$$

Proof. By Eq. 1 and Corollary 4, we know that:

$$\Pr[\mu_m > \Delta] \quad = \quad \Pr\left[\Delta < \frac{2\rho}{1-\rho}\mu_c\right]. \tag{3}$$

At time τ, the communication is re-enabled (cf. line 18 of Alg. 6). At this point in the execution, the probability that the number of blocks mined by each subgraph are within a $\pm\delta$ factor from their mean, is bound for $0 < \delta < 1$ and we have by Chernoff bounds [51], for $i \in \{1, 2\}$, the following:

$$\begin{cases} \Pr\left[X_i \geq (1+\delta)\mu_c\right] & \leq e^{-\frac{\delta^2}{3}\mu_c}, \\ \Pr\left[X_i \leq (1-\delta)\mu_c\right] & \leq e^{-\frac{\delta^2}{2}\mu_c}. \end{cases}$$

Thus, we can upper-bound the probability that the number of blocks mined by a subgraph diverges from its mean:

$$\Pr\left[|X_i - \mu_c| < \delta\mu_c\right] \quad > \quad 1 - 2e^{-\frac{\delta^2}{3}\mu_c}. \tag{4}$$

Observe that the probability that the two random variables X_1 and X_2 are both within $\pm\delta\mu_c$ is lower than the probability that their difference Δ is upper-bounded by $2\delta\mu_c$, hence we have:

$$\left(\Pr\left[|X_i - \mu_c| < \delta\mu_c\right]\right)^2 \leq \Pr[\Delta < 2\delta\mu_c].$$

It follows from Eq. 4 that:

$$\Pr[\Delta < 2\delta\mu_c] > \left(1 - 2e^{-\frac{\delta^2}{3}\mu_c}\right)^2. \tag{5}$$

Since $0 < \rho < \frac{1}{2}$ by line 7 of Alg. 6, we have $0 < \frac{\rho}{1-\rho} < 1$ so we fix $\delta = \frac{\rho}{1-\rho}$ that leads to the result. □

Theorem 7. *At time τ, the probability $\Pr[\mu_m > \Delta]$ that the expected number of blocks μ_m mined by the attacker is greater than the difference $\Delta = |X_1 - X_2|$ in blocks mined by the two subgraphs G_1 and G_2 is $\Pr[\mu_m > \Delta] > 1 - \varepsilon$ where $\varepsilon = 4e^{-\frac{\rho^2}{3(1-\rho)^2}\mu_c} = 4e^{-\frac{\rho^2}{3(1-\rho)^2}np}$.*

Proof. As $\mu_c \geq 0$ we have:

$$\begin{aligned} -\frac{\rho^2}{3(1-\rho)^2}\mu_c &\leq 0, \\ -e^{-\frac{\rho^2}{3(1-\rho)^2}\mu_c} &\geq -1, \end{aligned}$$

and we can apply Bernoulli's inequality (Fact 5) to Eq. 2 (Lemma 6):

$$\Pr[\mu_m > \Delta] > \left(1 - 2e^{-\frac{\rho^2}{3(1-\rho)^2}\mu_c}\right)^2 \geq 1 - 4e^{-\frac{\rho^2}{3(1-\rho)^2}\mu_c}. \tag{6}$$

From line 13 of Alg. 6, we know that $\tau \geq \frac{(1-\rho)6d\log(\frac{4}{\varepsilon})}{\rho^2 t}$ which leads to:

$$\begin{aligned} \tau &\geq \frac{(1-\rho)6d\log(\frac{4}{\varepsilon})}{\rho^2 t}, \\ \frac{(1-\rho)t\tau}{2d} &\geq \frac{3(1-\rho)^2\log(\frac{4}{\varepsilon})}{\rho^2}. \end{aligned}$$

By Eq. 1 we have:

$$\begin{aligned} \mu_c &\geq \frac{3(1-\rho)^2\log(\frac{4}{\varepsilon})}{\rho^2}, \\ \frac{\rho^2\mu_c}{3(1-\rho)^2} &\geq \log\left(\frac{4}{\varepsilon}\right), \\ -\frac{\rho^2\mu_c}{3(1-\rho)^2} &\leq \log\left(\frac{\varepsilon}{4}\right), \\ e^{-\frac{\rho^2\mu_c}{3(1-\rho)^2}} &\leq \frac{\varepsilon}{4}, \\ 1 - 4e^{-\frac{\rho^2}{3(1-\rho)^2}\mu_c} &\geq 1 - \varepsilon. \end{aligned}$$

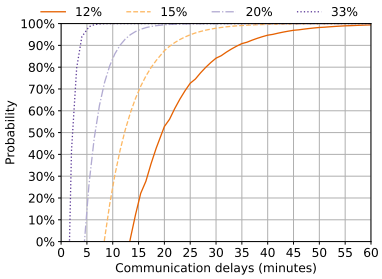
Replacing this expression in Eq. 6 leads to the result where ε can be tuned by the attacker by adjusting τ. □

Corollary 8. *A blockchain system that selects a main branch based on Nakamoto's consensus protocol, the* Ghost *protocol is corruptible.*

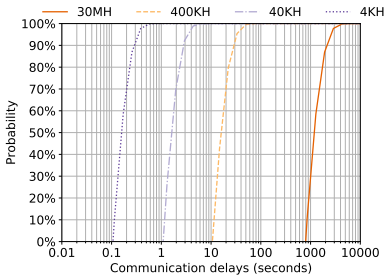
Proof. By examination of the code, we know that these protocols count the score according to the depth, num-desc and difficulty metrics to select one blockchain view and discard the other.

Number of nodes	50
Number of miners	15
Total mining power (MH/s)	20
Mining power of the most powerful miner (MH/s)	2.4
Difficulty (MH)	30

(a) The R3 settings used in the analysis as observed in June 2016



(b) Probability of the Balance attack in the R3 network as the communication delay increases for different portions of the mining power controlled by the attacker



(c) Probability of Balance attack in the R3 network (with 20 MH/s of total mining power, 12% of the mining power at the attacker) for different difficulties as the communication delay increases

Fig. 4. Simulation of the Balance attack with the difficulty of R3 and with the maximum mining power of R3

Since by Theorem 7, the expected metric score contributed by the attacker at time τ is greater than the difference Δ with probability $1-\varepsilon$, we know that the attacker can make the system discard the blockchain view of either G_1 or G_2 with probability $1-\varepsilon$ by contributing to the score of the other subgraph, hence making the blockchain system corruptible. □

5 ANALYSIS OF THE R3 NETWORK

In this section, we analyze the probability of applying the Balance attack to the R3 Ethereum testnet. R3 is a consortium of more than 50 banks that has tested blockchain systems and in particular Ethereum in a consortium private chain context over 2016 [63]. The statistics of the R3 network were gathered through the `eth-netstat` applications at the end of June 2016 by the R3 team who shared this information with us. The network consisted at that time of $|V| = 50$ nodes among which only 15 were mining. The mining power of the system was about 20 MH/s, the most powerful miner mined at 2.4 MH/s or 12% of the total mining power while the difficulty of the crypto-puzzle was observed close to 30 MH as summarized in Figure 4(a). One may think that 12% is a relatively large share for a single member in a 50-member consortium, however, it is important to note that the distribution of the blockchain decision power is typically more skewed, as three nodes own generally more than the rest of the systems in existing blockchains.[4] Today, one may use Ethereum with proof-of-authority to cope with the Balance attack, however, it has recently been vulnerable to another type of attack [29]. Since we communicated our Balance attack analysis to the R3 consortium, R3 has developed Corda, a crash fault tolerant distributed ledger [16].

5.1 How the most powerful node could double-spend

Let assume that the attacker is the `r3` node with $\rho = 12\%$ of the mining power as depicted in Figure 5 and that it delays communication between subgraphs G_1 and G_2, each with mining power $\frac{1-\rho}{2}t = 8.8$ MH/s. The probability p of solving the crypto-puzzle per hash tested is $\frac{1}{30\times10^6}$ so that the mean is $\mu_c = \frac{(1-\rho)t\tau}{2d} = \frac{8.8\times10^6\times1180}{2\times30\times10^6} = 346.13$ if we wait for 19 minutes and 40 seconds, i.e., 1180 seconds. The attacker creates, in expectation, a block every $\frac{30}{2.4} = 12.5$ seconds or

[4]On 20th June 2020, only 3 nodes own together more decision power than the rest of the system of more than 70% of the blockchains monitored at https://bitcoinera.app/arewedecentralizedyet/.

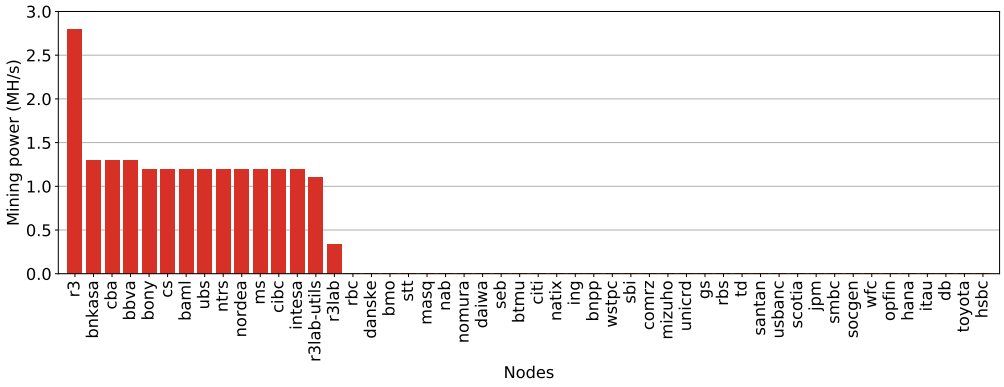


Fig. 5. The mining power of the R3 Ethereum miners as reported by `eth-netstats` to R3 and shared we us as of June 2016

$\lfloor \frac{1180}{12.5} \rfloor = 94$ blocks during the 19 minutes and 40 seconds. Hence let $\delta = \rho/(1-\rho) = 0.136$. The probability that the attack is a success is 53%.

5.2 A coalition of 33% of mining power needs a 4 minute delay to attack with 94% of success

Malicious nodes may have an incentive to form a coalition in order to exploit the Balance attack to double-spend. In this case, it allows the attacker to control a larger portion of the mining power of the system which, in turn, increases the chances of success of the Balance attack.

Let assume that the attacker controls $\rho = \frac{1}{3}$ of the total mining power, which represents $\rho t = 6.7$ MH/s. In this case the attacker delay communications between two subgraphs G_1 and G_2 with mining power of $\frac{1-\rho}{2}t = 6.7$ MH/s each. If we wait for 4 minutes, i.e., 240 seconds, then each isolated graph and attacker would mine $6.67 \cdot 10^6 \cdot 240/(30 \cdot 10^6) = 53.4$ blocks. The probability that the attack succeeds would become $1 - e^{-\rho^2/3(1-\rho)^2 \cdot 53.4}$, which is around 94%.

5.3 Tradeoff between communication delays and mining power

To illustrate the tradeoff between communication delay and the portion of the mining power controlled by the attacker, we consider the R3 network with a 30 MH total difficulty, a 20 MH/s total mining power and plot the probability as the communication delay increases for different portions of the mining power controlled by the attacker. Figure 4(b) depicts this result. As expected, the probability increases exponentially fast as the delay increases, and the higher the portion of the mining power is controlled by the attacker the faster the probability increases. In particular, in order to issue a balance attack with 90% probability, 35 minutes are needed for an attacker controlling 12% of the total mining power whereas only 11 minutes are sufficient for an attacker who controls 20% of the mining power. Our empirical analysis deferred to Section 7.3 reveals that the attack is actually more frequently successful, indicating that this theoretical analysis is conservative.

5.4 Tradeoff between communication delays and difficulties

Another interesting aspect of proof-of-work blockchains is the difficulty parameter d. As already mentioned, this parameter impacts the expected time it takes for a miner to succeed in solving the crypto-puzzle. When setting up a private chain, one has to choose a difficulty to make sure the miners would mine at a desirable pace. A too high difficulty reduces the throughput of the system without requiring leader election [32] or consensus sharding [48]. A too

low difficulty increases the probability for two correct miners to solve the crypto-puzzle before one can propagate the block to the other, a problem of Bitcoin that motivated the Ghost protocol [66].

Figure 4(c) depicts the probability of the Balance Attack when the communication delay increases for different difficulties without considering the time for a block to be decided. Again, we consider the R3 Ethereum network with a total mining power of 30 MH/s and an attacker owning $\rho = 12\%$ of this mining power and delaying communications between $k = 2$ subgraphs of half of the remaining mining power ($\frac{1-\rho}{2} = 44\%$) each. The curve labelled 4 KH indicates a difficulty of 4000 hashes, which is also the difficulty chosen by default by Ethereum when setting up a new consortium blockchain. This difficulty is dynamically adjusted by Ethereum at runtime to keep the mining block duration constant in expectation, however, this adaptation is dependent on the visible mining power of the system. The curve labelled 30 MH indicates the probability for the difficulty observed in the R3 Ethereum network. We can clearly see that the difficulty impacts the probability of the Balance attack. This can be explained by the fact that the deviation of the random variables $X_1, ..., X_k$ from their mean μ_c is bounded for sufficiently large number of mined blocks.

6 RUNNING THE BALANCE ATTACK ON A PRIVATE ETHEREUM BLOCKCHAIN

In this section, we experimentally produce the attack on an Ethereum private chain involving up to 18 physical distributed machines. To this end, we configure a realistic network with 15 machines dedicated to mining as in the R3 Ethereum network we described in Section 5, and 3 dedicated network switches. All experiments were run on 18 physical machines of the Emulab environment where a network topology was configured using ns/2 [58] as depicted in Figure 6. The topology consists of three local area networks configured through a ns/2 configuration file with 20 ms latency and 100 Mbps bandwidth. All miners run the `geth` [31] Ethereum client v.1.3.6 and the initial difficulty of the crypto-puzzle is set to 40 KH. The communication graph comprises the subgraph G_1 of 8 miners that includes the attacker and a subgraph G_2 of 7 correct miners.

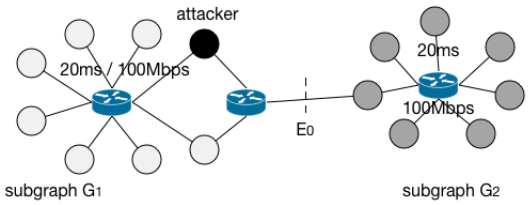


Fig. 6. The topology of our experiment involving 15 miners with subgraph G_1 including the attacker depicted in black and subgraph G_2 depicted in grey

6.1 Favoring one blockchain view over another

We run our first experiment for 2 minutes. We delayed the link E_0 for 60 seconds so that both subgraphs mine in isolation from each other during that time and end up with distinct blockchain views. After the delay we take a snapshot, at time t_1, of the blocks mined by each subgraphs and the two subgraphs start exchanging information normally leading to a consensus regarding the current state of the blockchain. At the end of the experiment, after 2 minutes, we take another snapshot t_2 of the blocks mined by each subgraph.

Table 3 lists the number of blocks (excluding uncles) of the blockchain views of G_1 and G_2 at time t_1, while the two subgraphs

Table 3. Number of blocks in the main branch (excluding uncles) mined by the subgraphs G_1 and G_2; the attacker influences the selection of branches and keeps blocks from G_1 but discards blocks from G_2

	# blocks at t_1	# blocks discarded at t_2	# blocks kept at t_2	retention
G_1	52	39	13	25%
G_2	58	58	0	0%

did not exchange their view, and at time t_2, after the subgraphs
exchanged their blocks. Note that we did not represent the uncle blocks to focus on the main branches. We observe that the blockchain view of the subgraph G_1 was adopted as the valid chain in contrast with the other blockchain view of the subgraph G_2. In particular, we retrieved 13 blocks of the main branch of G_1 at time t_1 in the main branch selected at t_2. As expected, all the blocks of G_2 at time t_1 were discarded from the main branch by time t_2.

6.2 Blocks mined by an attacker and two subgraphs

We now report the total number of blocks mined, especially focusing on the creation of uncle blocks. More precisely, we compare the number of blocks mined by the attacker against the difference of the number of blocks Δ mined by each subgraph. We know from the analysis that it is sufficient for the attacker to mine at least $\Delta + 1$ blocks in order to be able to discard one of the k blockchain views, allowing for double-spending. The experiment is similar to the previous experiment in that we also used Emulab with the same ns/2 topology, however, we did not introduce delays and averaged results over 10 runs of 4 minutes each.

Figure 7(a) depicts the minimum, maximum and average blocks obtained over the 10 runs. The vertical bars indicate minimum and maximum. First, we can observe that the average difference Δ is usually close to its minimum value observed during the 10 runs. This is due to having a similar total number of blocks mined by each subgraph in most cases with few rare cases where the difference is larger. As we can see, the total number of blocks (including uncles) mined during the experiment by the attacker is way larger than the difference in blocks Δ mined by the two subgraphs. This explains the success of the Balance attack as was observed in Section 6.1.

6.3 The role of forks

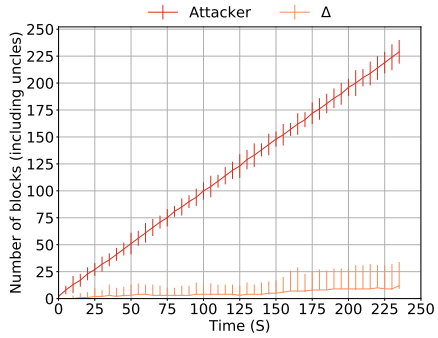
In the previous experiment, we focused on the total number of blocks without differentiating the blocks that are adopted in the main branch and the uncle blocks that are only part of the local blockchain views. Even though the Ethereum selection does not take the uncle blocks into account, the Ghost protocol accounts for these uncle blocks to decide the current state of the blockchain as we explained previously in Section 2.

Figure 7(b) indicates the number of uncle blocks in comparison to the blocks accepted on the current state of the blockchain for subgraphs G_1 and G_2, and the attacker (referred to as 'Malicious'). As expected, we can observe that the attacker does not produce any uncle block because the attacker mines the block in isolation of the rest of the network, successfully appending each mined block consecutively to the latest block of its current blockchain view. We note several uncle blocks in the subgraphs, as correct miners may mine blocks concurrently at the same indices.

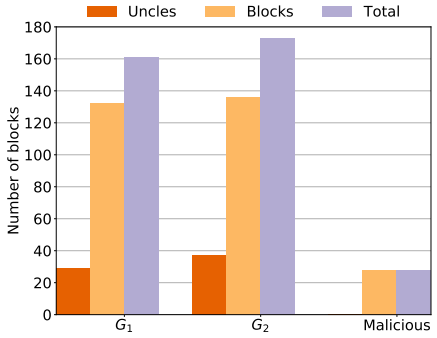
Figure 7(c) depicts the creation of the number of mined blocks (excluding uncle blocks) over time for subgraphs G_1 and G_2, and the attacker (Malicious). As we can see the difference between the number of blocks mined on the subgraphs is significantly smaller than the number of blocks mined by the attacker. This explains why the Balance attack was observed in this setting.

6.4 Relating connectivity to known blocks

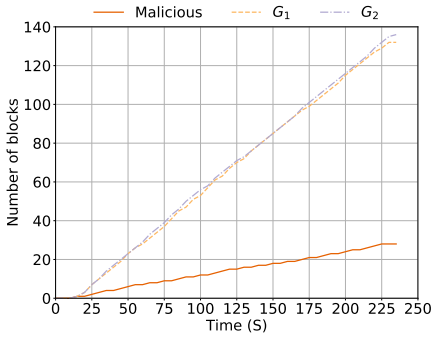
Figure 7(d) depicts the evolution of the number of blocks created by two subgraphs G_1 and G_2, where $|V_1| = |V_2| = 7$, as time elapses as well as the number of neigbours of two nodes p_1 and p_2 in each of these two subgraphs. For the first minute, the two subgraphs are disconnected in that the communication is artificially cut between the two subgraphs. We defer the explanation on how to obtain this temporary network partition using a BGP-hikacking attack to Section 7. During that time, the attacker selects a subgraph, in this case G_1, in which it mines blocks, contributing to the blockchain



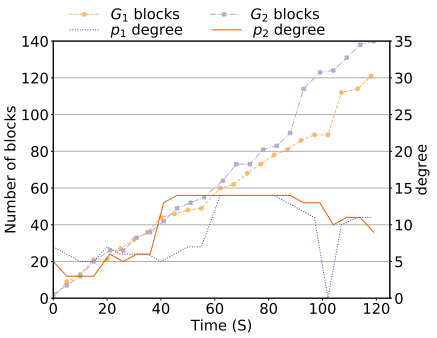
(a) Number of blocks mined by the attacker and the difference Δ in the number of blocks mined by the subgraphs



(b) Number of blocks mined by the attacker and the two subgraphs G_1 and G_2



(c) The depth of the blockchains mined by the attacker and two subgraphs G_1 and G_2



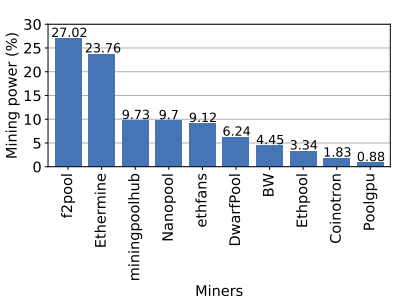
(d) Evolution of blocks and degrees where the attacker in G_1 delays the communication between G_1 and G_2

Fig. 7. Distributed experiments performed on a blockchain with 15 physical machines connected by a 100 Mbps network.

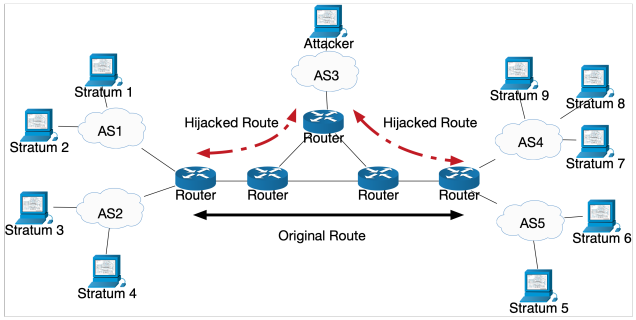
view of this subgraph. After about 60 seconds, the two subgraphs start to communicate and we observe an increase in the number of neighbors (or degree) to which p_1 and p_2 are connected. The earlier peak for p_2 than p_1 is due to a clock drift of less than 20 seconds we observe between the two machines as they were not synchronizing their clocks using NTP. As G_1 and G_2 exchange their respective blocks, we observe that the number of blocks increases more rapidly after 60 seconds than before 60 seconds. Upon reconnection, the subgraphs invoke the consensus protocol to select and adopt the correct chain. In this case, using the Ethereum consensus protocol, the chain mined by G_1 is chosen, to which the attacker contributed. This result reveals that the adoption of a chosen blockchain is plausible, given that the attacker is able to sufficiently delay messages between subgraphs.

7 RUNNING THE BALANCE ATTACK ON THE EMULATED PUBLIC ETHEREUM

In this section, we show experimentally how someone can double spend after partitioning Ethereum by hijacking BGP. To quantify the risk of a partitioning attack with the distribution of mining power in the main chain, we emulated the public Ethereum top-10 miners as observed on http://etherscan.io during one week on August 3^{rd}, 2017 in Figure 8(a). We deployed 10 virtual machines (VMs) linked through 5 *Border Gateway Protocol (BGP)* [40] routers, as shown in Figure 8(b), and controlled in our private cloud infrastructure via OpenStack.



(a) Top 10 Ethereum miners with their mining power as observed on http://etherscan.io from July 27^{th} to Aug. 3^{rd}, 2017



(b) Experimental topology for the emulated public Ethereum

Fig. 8. Experiment setup for the emulated public Ethereum

7.1 Adjusting the mining power of the nodes

To experience a realistic mining power distribution, we adjusted the mining power of our nodes to the mining power of the top-10 mining pools of Ethereum, which cumulatively represent more than 96% of the whole mining power. Mining pools are groups of miners that combine their computational power to mine blocks and share the rewards among themselves. They are appealing in public blockchains as they allow miners to receive a smaller yet more frequent reward than if they were mining individually.

To obtain the mining power distribution of miners retrieved in Figure 8(a) among our own VMs, we fixed the quantum of CPU time allocated to each machine using Linux `cgroups` [44]. Linux `cgroups` allow us to specify the CPU quota Q that a VM can consume within a period of time T. Given the same value of T, we vary Q on all virtual machines based on their correspondent mining power percentage of the miners. As a result, we obtained the proportion we listed in Table 4 close to 1 decimal.

7.2 Autonomous systems and BGP hijack

To create a realistic environment, the 10 virtual machines were divided into *Autonomous Systems (ASes)* as used in the public Internet for routing purpose. ASes are groups of networks under the control of a single technical administration [41]. ASes have their own routing policy for internal traffic but use BGP for dynamic inter-AS routing. The conventional way to perform routing attacks on the public Internet is to advertise the untruthful routes via a dynamic routing protocol, therefore the connectivity between ASes with direct peering mitigates the effect of routing attacks. Unfortunately, BGP does not incorporate a mechanism to check whether an origin AS owns the IP prefixes that it announces. This makes a protocol vulnerable to route hijacking.

We then combined a route hijacking attack with the Balance attack to evaluate the risks of double spending in Ethereum v1.5. First, the route hijacking is used to delay communication, then the Balance attack is used to turn these delays into double spending. To this end, we assign the role of the attacker to one of the miner in each of our attack instances. As indicated in Figure 8(b), the attacker takes control over one router to prevent AS1 and AS2 from communicating with AS4 and AS5 during 7 minutes. During that time, the attacker issues a transaction to one group and contributes to the block creation of the other group in order to discard its previously issued transaction. Since it is commonly recommended to wait for 12 confirmations or $m = 11$ to be confident about the immutability of a transaction since the version Homestead of Ethereum (cf. Table 1), we consider the double spending successful when the transactions contained in a block followed by 11 consecutive blocks gets discarded.

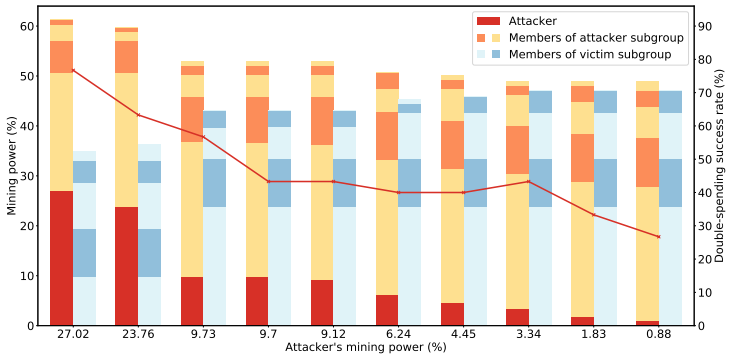


Fig. 9. Success of double spending with miners of similar power to the top 10 Ethereum miners from July 27^{th} to Aug. 3^{rd}, 2017

7.3 Risks of double spending after executing the balance attack

The goal of this experiment is to detect whether the attacker transaction is discarded due to the choice of the canonical branch. To this end, we inspected the blockchain after delaying the communication for 7 minutes in 30 consecutive runs and measured the average success of the attack. As indicated in Figure 9, we observe that only 10% of the mining power is sufficient for the double spending to be successful most of the time. With only 27% of the mining power, the success of the attack reaches 76%. Note that this attack success confirms the lower bound on the success probability obtained with our theoretical analysis in Section 5.3: this empirically observed attack success is higher than our lower bound.

7.4 On the difficulty of partitioning mining pools

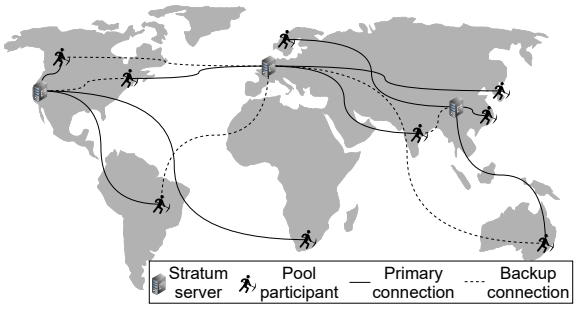
Although each mining pool is viewed as a centralized miner from the rest of the system, their connectivity to the system is different from the connectivity of a central miner as explained below. At the heart of each pool is a small number of *stratum servers*, which act as communication proxies between pool members and the rest of the blockchain network. Information from the blockchain network flows in and out the mining pool via the stratum servers. These servers coordinate the crypto-puzzle resolution by sending updates and distributing workload to pool members. This mechanism hides pool members behind the stratum servers, such that their information is not exposed to the blockchain network. Finally, a stratum server hides information of a pool member from one another, as it eliminates the need for direct communication among the members.

In order to gain some insights regarding the public Ethereum blockchain, we combined mining pools of the top-10 miners from Figure 8(a) with their network connectivity information in Table 4. To this end, to each named mining pool we registered a miner that could gather IP and *Autonomous System (AS)* information. In particular, we estimated locations of the servers by querying 5 geo-IP databases [2–6]. To reduce the inaccuracies of geo-locations, we extracted the location indicated in the majority of these databases. To retrieve the number and owner of each AS, we relied on [1] and [7], both sources are based on the `whois` service.

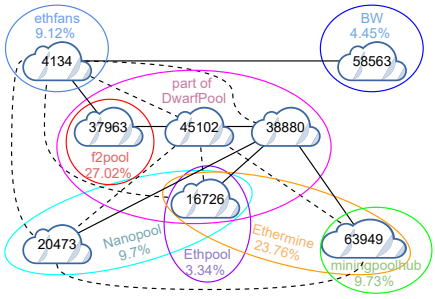
Table 4 lists the stratum servers of the top-10 Ethereum mining pools we retrieved. We noticed experimentally that if one of the stratum servers becomes unresponsive, then the corresponding miners would connect to the next stratum server they operate in order to remain connected to the pool. Hence, partitioning may result in having miners reconnect to a different AS. As an example, consider Figure 10(a), where a miner in India primarily connected to Europe (as indicated with a solid line) may reconnect to China (as indicated with the dashed line).

Table 4. Existing top 10 Ethereum mining pools with stratum servers, location and AS numbers from July 27^{th} to Aug. 3^{rd}, 2017

Pool Name	Stratum Servers	Location	ASN	AS Owner
f2pool	eth.f2pool.com	Hangzhou, China	37963	Alibaba (China) Technology Co., Ltd.
Ethermine	us1.ethermine.org	Montreal, Canada	16276	OVH SAS
	us2.ethermine.org	California, US	63949	Linode, LLC
	eu1.ethermine.org	France	16276	OVH SAS
	eu2.ethermine.org	France	16276	OVH SAS
	asia1.ethermine.org	Singapore	16276	OVH SAS
miningpoolhub	us-east.ethash-hub.miningpoolhub.com	Georgia, US	63949	Linode, LLC
	europe.ethash-hub.miningpoolhub.com	Hesse, Germany	63949	Linode, LLC
	asia.ethash-hub.miningpoolhub.com	Tokyo, Japan	63949	Linode, LLC
Nanopool	eth-eu1.nanopool.org	France	16276	OVH SAS
	eth-eu2.nanopool.org	France or Italy	16276	OVH SAS
	eth-asia1.nanopool.org	Singapore	16276	OVH SAS
	eth-us-east1.nanopool.org	Montreal, Canada	16276	OVH SAS
	eth-us-west1.nanopool.org	California, US	20473	Choopa, LLC
ethfans	guangdong-pool.ethfans.org	Fujian, China	4134	No.31,Jin-rong Street
	huabei-pool.ethfans.org	Fujian, China	4134	No.31,Jin-rong Street
DwarfPool	eth-eu.dwarfpool.com	France	16276	OVH SAS
	eth-us.dwarfpool.com	Montreal, Canada	16276	OVH SAS
	eth-us2.dwarfpool.com	Las Vegas, US	53667	FranTech Solutions
	eth-ru.dwarfpool.com	France	16276	OVH SAS
	eth-asia.dwarfpool.com	Taiwan	59253	Leaseweb Asia Pacific pte. ltd.
	eth-cn.dwarfpool.com	Shanghai, China	37963	Alibaba (China) Technology Co., Ltd.
	eth-cn2.dwarfpool.com	Beijing, China	37963	Alibaba (China) Technology Co., Ltd.
	eth-sg.dwarfpool.com	Singapore	59253	Leaseweb Asia Pacific pte. ltd.
	eth-au.dwarfpool.com	Melbourne, Australia	38880	Micron21 Melbourne Australia Datacentre
	eth-ru2.dwarfpool.com	Moscow, Russia	42632	MnogoByte LLC
	eth-hk.dwarfpool.com	Hong Kong	45102	Alibaba (China) Technology Co., Ltd.
	eth-br.dwarfpool.com	Sao Paulo, Brazil	262287	Maxihost Hospedagem de Sites Ltda
	eth-ar.dwarfpool.com	Rosario, Argentina	27823	Dattatec.com
BW	ether.bw.com	Wuhan, China	58563	CHINANET Hubei province network
Ethpool	us1.ethpool.org	Montreal, Canada	16276	OVH SAS
	us2.ethpool.org	Montreal, Canada	16276	OVH SAS
	eu1.ethpool.org	France	16276	OVH SAS
	asia1.ethpool.org	Singapore	16276	OVH SAS
Coinotron	coinotron.com	Poland	51290	HOSTEAM-AS
Poolgpu	eth.poolgpu.com	Hangzhou, China	37963	Alibaba (China) Technology Co., Ltd.



(a) The hypothetical connectivity of the stratum servers and pool miners of the Ethereum main chain. The miners are connected to stratum servers in different regions of the world as indicated by the solid lines. Upon a network attack like our BGP attack, the miners may reconnect to a different stratum server located in a different region following the dashed link. As these reconnections are hard to predict they may mitigate the success of the network attack depending on the AS of the stratum server they reconnect to.



(b) The inter-AS connections among the group of ASes that host stratum servers for public Ethereum mining pools. Each cloud represents an AS. An oval shape represents a mining pool. A thick line illustrates a link between two adjacent ASes or a peering connection, while a dash line indicates the existence of an indirect path between two ASes that requires only one transit AS in the middle.

Fig. 10. Hypothetical connectivity of stratum servers of the Ethereum main chain as well as the actual inter-AS connections among the group of ASes that host the stratum servers of the Ethereum main chain.

Table 5. Examples of other proof-of-work blockchains that use Nakamoto's consensus for fork resolution that may be susceptible to the attack from the top 15 Market Cap blockchains.

Blockchain	Target block time	Difficulty	Approx. market cap
Bitcoin	10 minutes	$16.5 \cdot 10^{12}$	$ 143,352,550,288
Ethereum	15 seconds	$2.2 \cdot 10^{15}$	$ 20,919,790,158
Bitcoin Cash	10 minutes	$506.0 \cdot 10^{9}$	$ 4,870,755,059
BitcoinSV	10 minutes	$384.5 \cdot 10^{9}$	$ 3,470,723,059
LiteCoin	2.5 minutes	$6.3 \cdot 10^{6}$	$ 3,087,326,181
Etherum Classic	15 seconds	$160.2 \cdot 10^{12}$	$ 757,774,257
Dogecoin	1 minute	$1.9 \cdot 10^{6}$	$ 263,832,426
Bitcoin Gold	10 minutes	$170 \cdot 10^{3}$	$ 158,089,267

In addition, it is more difficult to determine the precise proportion of mining power connected to each stratum server, again due to the numerous stratum servers each miner operates. Indeed, a mining pool identifier is nothing more than the wallet address to receive reward when a pool successfully mines a block. While it is possible to determine a block miner by examining header information, there is no way to pin down to the stratum server, as long as these servers put their reward into the same wallet address.

Second, the stratum servers typically hide the location of the mining pool participants, which makes it hard to isolate a group of pools of a specific mining power. In particular and as described in Figure 10(a), one cannot prevent a miner from India to reconnect to a stratum node located in China. Without information about the miners for a stratum server, one cannot guarantee the partition success of a network attack. It may *(i)* isolate a stratum server along with its miners completely, *(ii)* partition some miners, which reduces only a fraction of computational power from the pool, or *(iii)* cut off the connectivity between a stratum server and pool participants, such that those participants decide to reconnect to different stratum servers.

Third, BGP-hijacking cannot affect the direct interconnection between ASes, because ASes are aware of static network prefixes that belong to their peer ASes. Apart from exchanging routes at the *Internet Exchange Points (IXPs)*, any pair of ASes may decide to establish either layer 2 or layer 3 links to connect their networks directly. This prevents dynamic routing attacks like the BGP hijacking we discussed above in Section 7.2. To better understand the applicability of the attack to the Ethereum public blockchain, we retrieved the direct peering information of the 8 ASes we identified using available information [8] and listed these interconnections in Figure 10(b). Among the top-10 public Ethereum mining pools, 7 of them solely rely on this group of ASes; together, they account for more than 87% from the total mining power of the network. As the majority of ASes in this group are linked by direct peering, it appears extremely difficult to partition Ethereum's overlay. For example, `f2pool` may send and update to `ethfans` via a peering connection, which in turn forwards the update to `BW` via another peering connection. Without an attacker gaining access to configuration on the border routers of these ASes, it will remain difficult to partition a pool from the rest of the group.

8 APPLICATION TO OTHER BLOCKCHAINS AND COUNTER-MEASURES

Other proof-of-work blockchains that resolve forks with Nakamoto's consensus, Ethereum's consensus or the Ghost protocol, may be vulnerable to the Balance attack. Table 5 displays blockchains from the top 15 market cap that have a combination of proof-of-work and Nakamoto's consensus that are likely to be vulnerable to the Balance attack. The derivatives of Bitcoin, including Bitcoin Cash, Bitcoin Gold, Bitcoin Satoshi's Vision, all utilise a proof-of-work based block formation mechanism, and resolve forks using the Nakamoto's consensus longest chain. Although the block times may differ, the core principle of the longest chain makes them, in principle, vulnerable to the Balance attack.

For the same reasons, we do not exclude the possibility for other blockchains combining some of the three consensus algorithms we consider or some variants of them, to also be vulnerable to the Balance attack.

Although it is almost impossible to eliminate risks of double-spending, there is a range of counter-measures to mitigate the effect of the Balance attack. The simplest counter-measure is to increase the number of confirmations necessary to commit transactions. Not only will it require to extend the duration of the attack, but it will also increase the chance that the attack will be detected or disrupted by a number of configuration changes. The drawback of this counter-measure is the latency increase. The second counter-measure consists of selecting the peers to query the transaction status. Before delivering goods, a merchant could mitigate the risk by simply querying a transaction status from many miners including a group of nodes that are located further away in the network topology, such as two nodes in different ASes. This technique is already used in the industry with the Redbelly Blockchain [24, 68] to achieve Byzantine fault tolerance. The merchant should deliver the good only if the queried distant miners confirm that the transaction is indeed committed. A third counter-measure is to mitigate the risk of the man-in-the-middle-attack by leveraging multihomed ASes and multihomed hosts for a consortium and a private blockchain environment, respectively. For an attacker to perform a man-in-the-middle attack, it will have to control all network paths in use. The last counter-measure consists of running a blockchain that does not fork but that reaches agreement on a block before appending it. Despite several notable vulnerabilities in blockchain consensus protocols [70], blockchain technology has matured and there exist today blockchain consensus protocols that were formally verified with model checking [13, 14], hence drastically reducing the risks of errors. Such consensus is reached by n known participants, and rotation solutions already exist to prevent these participants from being bribed [67].

9 RELATED WORK

Traditional attacks against Bitcoin consist of waiting for some external action, like shipping goods, in response to a transaction before discarding the transaction from the main branch. As the transaction is revoked, the issuer of the transaction can reuse the coins of the transaction in another transaction. As the side effects of the external action cannot be revoked, the second transaction appears as a "double-spending".

Attacks based on mining power. Perhaps the most basic form of such an attack assumes that an application takes an external action as soon as a transaction is included in a block [12, 34, 45]. The first attack of this kind is called Finney's attack and consists of solo-mining a block with a transaction that sends coins to itself without broadcasting it before issuing a transaction that double-spends the same coin to a merchant. When the goods are delivered in exchange of the coins, the attacker broadcasts its block to override the payment of the merchant. The vector76 attack [73] consists of an attacker solo-mining after block b_0 a new block b_1 containing a transaction to a merchant to purchase goods. Once another block b'_1 is mined after b_0, the attacker quickly sends b_1 to the merchant for an external action to be taken. If b'_1 is accepted by the system, the attacker can issue another transaction with the coins spent in the discarded block b_1.

The attacks become harder if the external action is taken after the transaction is committed by the blockchain. Rosenfeld's attack [64] consists of issuing a transaction to a merchant. The attacker then starts solo-mining a longer branch while waiting for m blocks to be appended so that the merchant takes an external action in response to the commit. The attack success probability depends on the number m of blocks the merchant waits before taking an external action and the attacker mining power [36]. However, when the attacker has more mining power than the rest of the system, the attack, also called *majority hashrate attack* or *51-percent attack*, is guaranteed successful, regardless of the value m. To make the attack successful in expectation when the attacker owns only a quarter of the mining power,

the attacker can incentivize other miners to form a coalition [33] until the coalition owns more than half of the total mining power. Note that this would be insufficient to attack the ZLB recent blockchain [62].

Attacks against Bitcoin. Without a quarter of the mining power, discarding a committed transaction in Bitcoin requires additional power, like the control over the network. It is well known that delaying network messages can impact Bitcoin [25, 37, 38, 53, 56, 59, 66]. Decker and Wattenhofer already observed that Bitcoin suffered from block propagation delays [25]. Godel et al. [37] analyzed the effect of propagation delays on Bitcoin using a Markov process. Garay et al. [35] investigated Bitcoin in the synchronous communication setting, however, this setting is often considered too restrictive [23]. Heilman et al. [42] presented the Eclipse attack against one Bitcoin node. This attack consists of isolating one node from the rest of the system by controlling the machines it connects to. Pass et al. [59] extended the analysis for when the bound on message delivery is unknown and showed in their model that the difficulty of Bitcoin's crypto-puzzle has to be adapted depending on the bound on the communication delays. Saad et al. [65] incorporate network synchronization into the Bitcoin security model but their partition attack cannot last more than 10 minutes [39]. This series of work reveal an important limitation of Bitcoin: delaying propagation of blocks can waste the computational effort of correct nodes by letting them mine blocks unnecessarily at the same index of the chain. In this case, the attacker does not need more mining power than the correct miners, but simply needs to expand its local blockchain faster than the growth of the longest branch of the correct blockchain.

Some work already evoked the danger of using proof-of-work techniques in a consortium context [38]. In particular, experiments demonstrated the impossibility of ordering even committed transactions in an Ethereum private chain without exploring the impact of the network delay [53]. We go further by showing that forkable blockchains may suffer from corruptibility.

Attacks against Ethereum. The Balance attack already influenced several research results on the Ethereum blockchain. Wei et al. [74] show that an attacker can maintain forks for long enough to attack a blockchain. Similar to the Balance attack, the key to their attack is to delay messages for long enough and their analysis make use of similar theoretical arguments, however, we are not aware of any implementation of the attack. In particular, they did not provide the BGP-Hijacking attack we presented here. Marcus et al. [49] show that the eclipse attack can be applied against Ethereum despite our observation that Ethereum has better connectivity than Bitcoin if the attacker manages to establish 25 incoming TCP connections to an Ethereum node before this node can establish a single one. By contrast, the adversary can initiate the Balance attack at any time as it does not target a newly joining node.

More recently, Henningsen et al. eclipse an Ethereum node [43] with false friends. They require the adversary to control two hosts in distinct subnetworks. Parinya et al. [29] proposed the Attack of the Clones against the two proof-of-authority versions of Ethereum, as implemented in geth and parity/open ethereum. Similar to the Balance attack, it also balances the number of validator among two groups, but needs to clone an Ethereum node. Neu et al. [57] propose the concept of Ebb-and-Flow protocols using Gasper [22], a proposal to combine Ghost with voting. They demonstrate an attack in which an adversary selectively releases votes to strategically fork the chain to accommodate double spending. Other attacks affecting Ethereum consist of miners trying to increase their chances of getting a reward [79] but are less detrimental for the user than double spending.

Mitigations. Network attacks against Bitcoin and their mitigation strategies have already been proposed [9, 55, 71]. A first mitigation strategy called SABRE [9] consists of relaying blocks world-wide through a set of connections that are resilient to routing attacks like the BGP-hijacking attach we implemented. A second mitigation strategy, called

Erlay [55], consists of multiplying the number of connections. Both mitigations have been exclusively tested against Bitcoin. SABRE does not evaluate Ethereum. Erlay mentions that the mitigation could apply to Ethereum as well by showing through simulations that the bandwidth savings and latency would not be impacted by a higher transaction rate. A complete experimentation of the Balance attack and BGP-hijacking would help measuring the extent to which this mitigation could be effective in Ethereum. Tran et al. [71] propose the Erebus attack that utilises a malicious Autonomous System to reroute a Bitcoin node's peer connection in order to achieve a man-in-the-middle, and does not require hijacking BGP, going undetected in many systems. The attack carefully harvests IPs which are then used to flood the victim who will reconnect to a desired node through the adversarial Autonomous System. Such an attack could allow someone to execute a Balance attack despite the above mitigation strategies.

10 CONCLUSION

This paper presents the Balance attack, an attack leveraging both mining power and connectivity to double spend on prominent forkable blockchain protocols such as Bitcoin and Ethereum and variants of these. The attack allows an attacker with low mining power to convince subgroups of honest nodes of a correct chain, leading to the eventual disregard of blocks and double-spending.

In this paper, we demonstrate that the attack leads to double spending with high probability and we quantify the probability of success in an existing consortium environment. We also confirm the attack empirically by hijacking BGP to reroute the traffic between subgroups of similar mining power. The success observed empirically confirms our theoretical lower bound on the probability of success of the Balance attack.

REFERENCES

[1] "CAIDA: Center for Applied Internet Data Analysis."
[2] "DB-IP - IP Geolocation and Network Intelligence." [Online]. Available: https://db-ip.com/
[3] "IP Address Details - ipinfo.io." [Online]. Available: http://ipinfo.io/
[4] "IP Address Geolocation to trace Country, Region, City, ZIP Code, etc." [Online]. Available: https://www.eurekapi.com/
[5] "IP Address to Identify Geolocation Information." [Online]. Available: http://www.ip2location.com/
[6] "IP Geolocation and Online Fraud Prevention | MaxMind." [Online]. Available: https://www.maxmind.com/en/home
[7] "Merit RADb." [Online]. Available: http://www.radb.net/
[8] "The CAIDA AS Relationships Dataset," Aug. 2017. [Online]. Available: http://www.caida.org/data/as-relationships/
[9] M. Apostolaki, G. Marti, J. Müller, and L. Vanbever, "SABRE: protecting bitcoin against routing attacks," in *26th Annual Network and Distributed System Security Symposium, NDSS*. The Internet Society, 2019.
[10] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: Large-scale network attacks on cryptocurrencies," arXiv, Tech. Rep. 1605.07524, 2016.
[11] ——, "Hijacking bitcoin: Routing attacks on cryptocurrencies," in *IEEE S&P 2017*, 2017, pp. 375–392.
[12] T. Bamert, C. Decker, L. Elsen, R. Wattenhofer, and S. Welten, "Have a snack, pay with bitcoins," in *13th IEEE International Conference on Peer-to-Peer Computing, IEEE P2P 2013, Trento, Italy, September 9-11, 2013, Proceedings*, 2013, pp. 1–5.
[13] N. Bertrand, V. Gramoli, I. Konnov, M. Lazic, P. Tholoniat, and J. Widder, "Brief announcement: Holistic verification of blockchain consensus," in *ACM Symposium on Principles of Distributed Computing (PODC)*, A. Milani and P. Woelfel, Eds. ACM, 2022, pp. 424–426.
[14] ——, "Holistic verification of blockchain consensus," in *36th International Symposium on Distributed Computing (DISC)*, ser. LIPIcs, C. Scheideler, Ed., vol. 246. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, pp. 10:1–10:24.
[15] A. Black, "Hashcash - a denial of service counter-measure," Cypherspace, Tech. Rep., 2002. [Online]. Available: http://www.hashcash.org/papers/hashcash.pdf
[16] R. G. Brown, J. Carlyle, I. Grigg, and M. Hearn, "Corda: An introduction," 2016. [Online]. Available: https://www.corda.net/
[17] V. Buterin, "Ethereum: A next-generation smartcontract and decentralized application platform," 2013.
[18] ——, "On slow and fast block times," 9 2015, https://blog.ethereum.org/2015/09/14/on-slow-and-fast-block-times/.
[19] ——, "How should i handle blockchain forks in my dapp?" 1 2016, https://ethereum.stackexchange.com/questions/183/how-should-i-handle-blockchain-forks-in-my-dapp/203/#203.
[20] ——, "Cbc casper and serenity," in *Community Ethereum Development Conference*, April 2019.
[21] ——, "Ethereum whitepaper," June 2020, https://ethereum.org/whitepaper/.

[22] V. Buterin, D. Hernandez, T. Kamphefner, K. Pham, Z. Qiao, D. Ryan, J. Sin, Y. Wang, and Y. X. Zhang, "Combining GHOST and casper," arXiv, Tech. Rep. 2003.03052, 2020. [Online]. Available: https://arxiv.org/abs/2003.03052

[23] C. Cachin, "Distributing trust on the internet," in *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, 2001, pp. 183–192.

[24] T. Crain, C. Natoli, and V. Gramoli, "Red belly: a secure, fair and scalable open blockchain," in *Proceedings of the 42nd IEEE Symposium on Security and Privacy (S&P'21)*, 2021.

[25] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *Proc. of the IEEE International Conference on Peer-to-Peer Computing*, 2013.

[26] D. Dolev and A. C. Yao, "On the security of public key protocols," *IEEE Trans. Inf. Theory*, pp. 198–208, 1983.

[27] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," *J. ACM*, vol. 35, no. 2, Apr. 1988.

[28] P. Ekparinya, V. Gramoli, and G. Jourjon, "Impact of man-in-the-middle attacks on ethereum," in *Proceedings of the 37th IEEE International Symposium on Reliable Distributed Systems (SRDS'18)*, Oct 2018. [Online]. Available: http://gramoli.redbellyblockchain.io/web/doc/pubs/ethereum-mitm-attacks.pdf

[29] ——, "The Attack of the Clones against Proof-of-Authority," in *Proceedings of the Network and Distributed Systems Security Symposium (NDSS'20)*, Feb 2020.

[30] Ethereum, "Ethereum network status," https://ethstats.net/, 2014, accessed: 2017-11-12.

[31] ——, "Geth - ethereum/go-ethereum," https://github.com/ethereum/go-ethereum/wiki/geth, 2017, accessed: 2018-03-20.

[32] I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2016.

[33] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *Proceedings of the 18th Int'l Conference Financial Cryptography and Data Security (FC)*, 2014, pp. 436–454.

[34] H. Finney, "Finney's attack," 2011. [Online]. Available: https://bitcointalk.org/index.php?topic=3441.msg48384#msg48384

[35] J. A. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *34th Annual Int'l Conf. on the Theory and Applications of Crypto. Techniques*, 2015, pp. 281–310.

[36] 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 (CCS)*, 2016, pp. 3–16.

[37] J. Göbel, H. Keeler, A. Krzesinski, and P. Taylor, "Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay," *Performance Evaluation*, July 2016.

[38] V. Gramoli, "On the danger of private blockchains," in *Workshop on Distributed Cryptocurrencies and Consensus Ledgers (DCCL'16)*, 2016.

[39] J. Ha, S. Baek, M. Tran, and M. S. Kang, "On the sustainability of bitcoin partitioning attacks," in *27th International Conference on Financial Cryptography and Data Security (FC)*, 2023-05.

[40] S. Hares, Y. Rekhter, and T. Li, "A Border Gateway Protocol 4 (BGP-4)," Jan. 2006.

[41] J. Hawkinson and T. Bates, "Guidelines for creation, selection, and registration of an Autonomous System (AS)," Mar. 1996.

[42] 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.

[43] S. A. Henningsen, D. Teunis, M. Florian, and B. Scheuermann, "Eclipsing ethereum peers with false friends," in *2019 IEEE European Symposium on Security and Privacy Workshops, EuroS&P Workshops 2019, Stockholm, Sweden, June 17-19, 2019*. IEEE, 2019, pp. 300–309. [Online]. Available: https://doi.org/10.1109/EuroSPW.2019.00040

[44] T. Heo, "Control Group v2," Oct. 2015. [Online]. Available: https://www.kernel.org/doc/Documentation/cgroup-v2.txt

[45] G. Karame, E. Androulaki, and S. Capkun, "Two bitcoins at the price of one? double-spending attacks on fast payments in bitcoin," *IACR Cryptology ePrint Archive*, vol. 2012, p. 248, 2012.

[46] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982.

[47] P. Litke and J. Stewart, "BGP hijacking for cryptocurrency profit," August 2014. [Online]. Available: https://www.secureworks.com/research/bgp-hijacking-for-cryptocurrency-profit

[48] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 17–30.

[49] 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. [Online]. Available: http://eprint.iacr.org/2018/236

[50] A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, and B. Bhattacharjee, "Discovering Bitcoin's network topology and influential nodes," University of Maryland, Tech. Rep., 2015.

[51] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 1995.

[52] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," 2008, http://www.bitcoin.org.

[53] C. Natoli and V. Gramoli, "The blockchain anomaly," in *Proceedings of the 15th IEEE International Symposium on Network Computing and Applications (NCA'16)*, Oct 2016.

[54] ——, "The balance attack or why forkable blockchains are ill-suited for consortium," in *47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2017, Denver, CO, USA, June 26-29, 2017*, 2017, pp. 579–590.

[55] G. Naumenko, G. Maxwell, P. Wuille, A. Fedorova, and I. Beschastnikh, "Erlay: Efficient transaction relay for bitcoin," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS*, L. Cavallaro, J. Kinder, X. Wang, and J. Katz, Eds. ACM, 2019, pp. 817–831.

[56] K. Nayak, S. Kumar, A. Miller, and E. Shi, "Stubborn mining: Generalizing selfish mining and combining with an eclipse attack," in *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, 2016, pp. 305–320.

[57] J. Neu, E. N. Tas, and D. Tse, "Ebb-and-flow protocols: A resolution of the availability-finality dilemma," in *2021 IEEE Symposium on Security and Privacy, SP 2021, May 24-27, 2021*. IEEE, 2021.

[58] NS-2, "Ns-2 user information - nsnam," http://nsnam.sourceforge.net/wiki/index.php/User_Information, 2011, accessed: 2018-03-20.

[59] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," Cryptology ePrint Archive, Tech. Rep. 454, 2016.

[60] B. Project, "Some things you need to know," 2020, https://bitcoin.org/en/you-need-to-know.

[61] R3, "R3," https://www.r3.com/, 2016, accessed: 2018-03-20.

[62] A. Ranchal-Pedrosa and V. Gramoli, "Zlb: A blockchain to tolerate colluding majorities," in *Proceedings of the 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2024.

[63] P. Rizzo, "R3 publishes vitalik buterin report on ethereum for banks," https://www.coindesk.com/r3-ethereum-report-banks/, 2016.

[64] M. Rosenfeld, "Analysis of hashrate-based double-spending," 2012.

[65] M. Saad, S. Chen, and D. Mohaisen, "Syncattack: Double-spending in bitcoin without mining power," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2021, pp. 1668–1685.

[66] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, 2015, pp. 507–527.

[67] D. Tennakoon and V. Gramoli, "Blockchain proportional governance reconfiguration: Mitigating a governance oligarchy," in *23rd IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 2023, pp. 545–556.

[68] D. Tennakoon, Y. Hua, and V. Gramoli, "Smart redbelly blockchain: Reducing congestion for web3," in *International Parallel and Distributed Processing Symposium (IPDPS)*, 2023, pp. 940–950.

[69] "The balance attack against proof of work blockchains," https://www.reddit.com/r/ethereum/comments/5rcm4o/the_balance_attack_against_proof_of_work/.

[70] P. Tholoniat and V. Gramoli, *Formal Verification of Blockchain Byzantine Fault Tolerance*. Springer, 2022, pp. 389–412. [Online]. Available: https://doi.org/10.1007/978-3-031-07535-3_12

[71] M. Tran, I. Choi, G. J. Moon, A. V. Vu, and M. S. Kang, "A stealthier partitioning attack against bitcoin peer-to-peer network," in *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*. IEEE, 2020, pp. 894–909. [Online]. Available: https://doi.org/10.1109/SP40000.2020.00027

[72] M. Tran, A. Shenoi, and M. S. Kang, "On the routing-aware peering against network-eclipse attacks in bitcoin," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021.

[73] vector76, "The vector76 attack," 2011. [Online]. Available: https://bitcointalk.org/index.php?topic=36788.msg463391#msg463391

[74] P. Wei, Q. Yuan, and Y. Zheng, "Security of the blockchain against long delay attack," in *Advances in Cryptology – ASIACRYPT 2018*, T. Peyrin and S. Galbraith, Eds. Cham: Springer International Publishing, 2018, pp. 250–275.

[75] J. Wilcke, "The ethereum network is currently undergoing a dos attack," https://blog.ethereum.org/2016/09/22/ethereum-network-currently-undergoing-dos-attack/, 2016, accessed: 2017-11-13.

[76] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger final draft - under review," 2014.

[77] ——, "Ethereum: A secure decentralised generalised transaction ledger," 2015, yellow paper. [Online]. Available: http://gavwood.com/paper.pdf

[78] ——, "Ethereum: A secure decentralised generalised transaction ledger final draft - under review," 2020, https://ethereum.github.io/yellowpaper/paper.pdf.

[79] A. Yaish, G. Stern, and A. Zohar, "Uncle maker: (time)stamping out the competition in Ethereum," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2023, pp. 135–149.