How to Pick Your Friends
A Game Theoretic Approach to P2P Overlay Construction

Saar Tochner
School of Computer Science and Engineering,
The Hebrew University of Jerusalem
saart@cs.huji.ac.il

Aviv Zohar
School of Computer Science and Engineering,
The Hebrew University of Jerusalem
avivz@cs.huji.ac.il

ABSTRACT

A major limitation of many blockchain systems is the lack of strong identities in the underlying P2P network. This allows any nodes to attack the system by creating multiple false personas, thereby disrupting the network's connectivity and sabotaging its operation. In this paper, we focus on P2P networks, and explore practical ways to defend them from such attacks. To do so, we employ a game theoretic approach to the management of each peer's list of known nodes and to the overlay construction mechanisms that utilize this list. We consider the interaction between the defender and attacker as a game. We show that the cost of attacks can be driven up substantially if the defender utilizes available information about peers it chooses to connect to, such as their IP address. In addition to theoretical analysis of the underlying game, we apply our approach to the Bitcoin P2P network and derive effective and practical strategies that guarantee a high safety level against attacks.

CCS CONCEPTS

• **Networks** → *Network economics*; **Network security**.

KEYWORDS

Cryptocurrencies; Eclipse Attack; Sybil Attack; Game Theory Analysis

1 INTRODUCTION

Blockchain systems, such as Bitcoin [15] and Ethereum [23], are utilizing their communication layer using P2P network. Unfortunately, these networks are extremely vulnerable to isolation attacks in which individual nodes that wish to participate in the network connect *only* to attackers and are effectively quarantined from all other honest participants. Attackers can then distort the view of nodes regarding events in the network and filter, change, or delay their messages at will.

Attacks of this sort give attackers great power. In many blockchain systems, and in particular the Bitcoin protocol, the consequences of isolating nodes can be devastating to the security of the network. A successful attack can manipulate the knowledge on the current blockchain of the isolated nodes, subvert their computational power to attack the rest of the network, and can apply censorship on issuing transactions.

In this paper we seek to make the attacker's job more difficult by suggesting strategies for the nodes in the P2P network to successfully avoid being quarantined, unless the attacker invests a great deal of resources. In particular, our work models the interaction between the attacker and the defender as a game. The strategy space of the attacker is to choose the nodes to corrupt in the network (at a cost per node), while the defender's strategy space consists of the different sets of nodes he can connect to. Our model leads to *practical* policies: We suggest ways to manage the limited buffer from which nodes choose connections. Our approach is related in spirit to security games: We consider the attacker and defender as rational nodes, with economical incentives, that seek to maximize their utility.

The main idea that we utilize to increase the cost for the attacker is to take advantage of properties of the connections to peers to try and pick connections that the attacker would be unlikely to control all at once. In the main example, we take inspiration from Bitcoin's P2P formation and consider the IP subnet mask of the peers. Our main assumption is that once an attacker purchases an IP address in some range of IPs, it is cheap to gain access to other similar IPs. Our peer-selection strategy in this case will tend to be biased towards selecting peers from many different IP ranges. This affects the way we should manage the IP addresses' buffer. One more important contribution that we make, is the application of a bloom filter to stop attackers from re-publishing addresses that have already been discarded from the IP address buffer. This allows us to overcome one of the main strategies attackers utilize in attacks: get their nodes to be over-represented in the victim's list of known nodes.

Then, our main practical result (Theorem 4) states that we can implement a safety-level strategy for the game, using only limited memory, in which the IPs are stored. Our contribution is thus a highly practical one.

Current P2P network formation The mechanisms through which P2P nodes choose their connections can vary, but most use the following general technique: Once an initial connection is established with a node (usually with the aid of a centralized server or hard-coded addresses that helps to bootstrap the process), nodes share the IP addresses of others with their connections, and maintain lists of possible peers to connect to. Such potential connections are stored in buffers and are often exchanged to keep the lists populated with "fresh" addresses. Therefore, the core two decisions to be made are: (i) when new IPs are announced: how to select which old IPs to evict from a full buffer, and (ii) when wishing to connect

AFT '20, October 21–23, 2020, New York, NY, USA

ACM ISBN 978-1-4503-8139-0/20/10…$15.00
https://doi.org/10.1145/3419614.3423252

to nodes, how to select which ones to connect to from IPs stored in the buffer.

A naive approach is to form connections to uniformly selected nodes in the buffer (as such random graphs are typically well connected) and to evict nodes uniformly at random from an overflowing buffer. Other approaches involve removing the oldest IPs, assuming that they are the ones most likely to be stale.

The susceptibility of naive strategies Attackers find it cheap to replace all the IPs in the buffer of the victim with IPs of attacker nodes (or with un-assigned IP addresses). The defender then fails to connect to other honest nodes and is quarantined by the attacker.

There are two approaches to an isolation attack (that are often combined together): Sybil attacks [4], in which the attacker creates many identities to increase his chance of getting connections; and Eclipse attacks [20], wherein the attacker increases the visibility of each of his nodes according to the peers' selection algorithm (e.g., advertising its own addresses more aggressively). If the defender receives many addresses belonging to the attacker, and naively evicts nodes from his buffer in order to store these new addresses, then the buffer will be overrun and the node will not be able to connect to any honest peers.

Bitcoin employs a more sophisticated eviction strategy, sorting IP addresses into buckets by combining the IP of the sender node with the IP address in the message itself (the node throughout this message arrived). Recent work shows this mechanism to be susceptible as well, as it still allows an attacker to overrun the buffer easily [8, 14].

The remainder of the paper is structured as follows: In the next section we briefly review related work, then in Section 2 we define utility functions for the players, the strategy space and the nodes' memory buffer. Section 3 shows results for the game without any restrictions on the buffer size. In Section 4 we restrict the size of the memory buffer, derive the Nash equilibrium strategies for the defender, and describe the algorithms to do so. Section 5 addresses a specific implementation of the cost function and contains our main practical theorem. Finally, Section 6 suggests a practical strategy the could be implemented on the Bitcoin network, evaluates it using real data that we crawled, and compares the result to other benchmarks. We conclude and discuss future work in Section 7.

1.1 Related Work

Douceur [4] was the first to expose the problem of multiple identities (Sybils) in P2P systems lacking strong identities. Many such systems have since been shown to be vulnerable to such attacks [9, 16, 22].

Bitcoin, a P2P currency system [15], was designed to work as an open system with no strong identities. Many attacks on Bitcoin are known and well researched. Some of them are time based [17], involve selfish mining behavior [6, 19], or undermine the operation of mining pools [5]. Multiple surveys provide a wide overview [2, 3, 12]. In this paper, we discuss the susceptibility of the overlay formation to attacks, such as [8, 21, 27, 30]. Ethereum, the second largest cryptocurrency today, had been found to be vulnerable to similar attacks as well [14, 24]. Botnets are often structured like P2P networks, and their susceptibility to such attacks has also been used against them [10, 11].

One approach to defend from Sybil attacks was to utilize a social network to form the peers' connections [25, 26] (Sybils are assumed to have few connections to honest participants in this setting). Unfortunately, most settings (and specifically Bitcoin) do not have this additional network of relations among peers to use for overlay formation. Another approach is to make it hard to create many entities by having to solve a computational puzzle [29], but this could be difficult to accomplish in different networks, if the nodes lack computational power (e.g., mobile devices, IOT, etc.). In this paper, we do not assume any external social relations nor do we require peers to provide proof of work.

The approach of modeling the interaction between attackers and defenders using game theoretic tools is well established [13, 28]. A series of papers on security games deals with several variants of such problems. These include the ARMOR project for security at airports [18], resilience to cyberphysical control systems [31], and information security [7].

2 MODEL

We assume the attacker wishes to separate the defender node from all other honest nodes. We model this as a game between a single defender and the attacker. The first model that we examine assumes (unrealistically) that the defender knows all the nodes in the "universe" V and is unrestricted by memory considerations. We later use this as a building block for the second model in which the defender has a limited knowledge and a restricted memory buffer.

It is important to note that (i) our model follows the selfishness approach of blockchain systems, and thus it was created to defend from attacks on a single node. This is not the general case, where the attacker may try to isolate a subset of nodes. We show a practical evaluation to this scenario in Section 6 (ii) In this paper we will approximate an honest node as "safe" simply if it has a single honest connection to the network (in this case, the node is not isolated and therefore can communicate with the network). (iii) An attack from unrestricted attacker is unpreventable.

2.1 Preliminaries

Let S^1, S^2 be the strategy sets of the defender and the attacker, respectively. Below we define $\mathcal{U} : S^1 \times S^2 \to \mathbb{R}$ as the defender's utility in the game (the attacker's utility is $-\mathcal{U}$, this is a 2-player zero-sum game).

Recall that (i) a strategy $s^1 \in S^1, s^2 \in S^2$ is called *Nash equilibrium* if $\forall s'^1 \in S^1, s'^2 \in S^2$ it holds that $\mathcal{U}(s'^1, s^2) \leq \mathcal{U}(s^1, s^2) \leq \mathcal{U}(s^1, s'^2)$ and (ii) a strategy $s^1 \in S^1$ is called an *L-safety-level strategy* iff $\forall s^2 \in S^2$, $\mathcal{U}(s^1, s^2) \geq L$ (a similar definition can be stated for player 2).

Denote V as the set of peers (either honest or those owned by the attacker) in the network. Let H be the number of connections the defender creates (as in Bitcoin implementation, we use a configurable constant. The default in Bitcoin is 8).

We further assume that the attacker gains some absolute value W_{att} from a successful attack against the defender (and that the defender suffers this as a loss). We will use this value as the reward in the game's utility function.

2.1.1 Cost Functions. We assume the attacker can corrupt or *acquire* a subset of nodes $B \subseteq V$ at a cost that we denote by $C(B)$,

where $C : 2^V \to \mathbb{R}_{\geq 0}$. This assumption is the basis of our perspective, and we will build the utility of the game around it.

Naturally, there are many possible ways to define such functions: constant cost (naive), costs that are relative to the round trip time, PoW (relative to the computation power), etc.

We start the discussion with the general cost function, where we show our theoretical results. After that, in Section 5, we choose a concrete cost function, conclude the main practical theorem, and evaluate the results. This concrete function is the IP-prefix cost function, in which acquiring an IP range costs c_{new} and every IP from that range costs c_{node}, therefore

$$C(B) = c_{new} \cdot number\_of\_masks\_in\_B + c_{node} \cdot |B|$$

2.2 The Limitless-Buffer Game

The 2-player zero-sum game between attacker and defender is defined as follows: The defender's strategy space S^1 contains possible sets of nodes that he may connect to: $S^1 = \{A \mid A \subset V, |A| = H\}$. The attacker's strategy space S^2 contains subsets of the universe V that he chooses to corrupt: $S^2 = \{B \mid B \subset V\} = 2^V$. The attack is considered successful if and only if the attacker owns *all* the nodes that the defender selected. The utility function is thus:

$$\mathcal{U}(A, B) = \begin{cases} C(B) & \text{if } A \not\subseteq B \\ C(B) - W_{att} & \text{if } A \subseteq B \end{cases}$$

We will usually consider the game with mixed-strategies $\sigma^1 \in \Delta_{S^1}, \sigma^2 \in \Delta_{S^2}$:

$$\mathcal{U}(\sigma^1, \sigma^2) = \sum_{B \in S^2} \sigma_B^2 C(B) - W_{att} \cdot \sum_{B \in S^2, A \in S^1} \sigma_A^1 \sigma_B^2 \delta_{A \subset B}$$

in the above σ_U^k is the probability that player k chooses the subset $U \in S^k$, and $\delta_{A \subset B}$ is 1 if $A \subset B$, and 0 otherwise.

The utility function thus represents the expected cost of an attack with regards to the mixed strategy. The attacker tries to minimize it (negative values are his reward), and the defender tries to increase its value.

2.3 The Restricted-Buffer Scenario

We wish to defend P2P networks in realistic settings so we must take buffers and limited knowledge of the world into account. We thus define the following refinement of the game: Assume that at any point in time the defender can only maintain a set of potential peers in his buffer. Let $\mathcal{B}$ denote the buffer size (the number of nodes for which information can be saved). The node receives a stream of announcements about nodes from which it selects ones to store in the buffer. We assume that the protocol demands that each node advertise itself at least once in each time period of length $\mathcal{T}$. Furthermore, we assume that attacker nodes may advertise themselves more frequently, as is often the case during eclipse attacks. If a node's details are stored in the buffer, and the buffer is full, a different stored record must be evicted first. The node then chooses its connections from the set of stored addresses.

Each node should specify an algorithm $\mathscr{R}$ that decides which records to save, based only on the cost to corrupt various peers in the network. As the defender, we try to find $\mathscr{R}$ that minimizes the expected utility of attackers regardless of the strategy that they adopt (i.e., a safety level).

The next section will discuss the game without the restricted-buffer constraint, and we will consider later, in Section 4.

3 ANALYSIS WITH AN UNRESTRICTED BUFFER

In this section we show results for the setting in which the nodes in V are known to all. The following theorem shows that it is better for the defender to err on the side of caution and over-estimate the damage from an attack, in case it is unknown:

Theorem 1. *Let $\mathcal{U}_{W_i}$ be the utility of the game in which a successful attack causes W_i damage. If strategy σ^1, is a l-safety-level for the defender in the game with W_1 and $W_2 \leq W_1$, then σ^1 is a l-safety level for the defender in $\mathcal{U}_{W_2}$, and generally:*

$$\forall \bar{\sigma}^2 \quad \mathcal{U}_{W_2}(\sigma^1, \bar{\sigma}^2) \geq \mathcal{U}_{W_1}(\sigma^1, \bar{\sigma}^2) \geq l$$

Proof. $\mathcal{U}_{W_2}(\sigma^1, \bar{\sigma}^2) =$

$$\sum_{B \in S^2} \bar{\sigma}_B^2 C(B) - W_2 \cdot \sum_{B \in S^2, A \in S^1} \sigma_A^1 \bar{\sigma}_B^2 \delta_{A \subset B} \geq$$

$$\sum_{B \in S^2} \bar{\sigma}_B^2 C(B) - W_1 \cdot \sum_{B \in S^2, A \in S^1} \sigma_A^1 \bar{\sigma}_B^2 \delta_{A \subset B}$$

$$= \mathcal{U}_{W_1}(\sigma^1, \bar{\sigma}^2) \geq l$$

The first step is directly from the definition, the second is since $W_2 \leq W_1$, and the last step is because σ^1 is a l-safety-level in $\mathcal{U}_{W_1}$. □

Next, we consider the attacker's strategies in Nash equilibria. We show that an attack may come in one of two forms: either the attacker corrupts no nodes at all or he places some small probability to "cover" *every* node.

Lemma 3.1. *For any Nash equilibrium (σ^1, σ^2) in the game it holds that either $\forall B$ that the attacker acquires, $\sigma_B^2 = 0$, or alternatively, for every node $v \in V$, there exists a subset B that contains it (i.e. $v \in B$), that the attacker chooses with some probability $\sigma_B^2 \neq 0$.*

Proof. Mark the defender's strategy to play pure $A \subset V$ with S_A^1. For a given $v \in V$, let us now assume that there exists $A \in S^1$ s.t. $v \in A$ and for all B with $A \subseteq B$ it holds that $\sigma_B^2 = 0$, then the attacker never answers the defender's strategy S_A^1, therefore $\mathcal{U}(S_A^1, \sigma^2) \geq 0$. But it follows by the Nash equilibrium that $\mathcal{U}(\sigma^1, \sigma^2) \geq \mathcal{U}(S_A^1, \sigma^2)$. Clearly $0 = \mathcal{U}(\sigma^1, 0) \geq \mathcal{U}(S_A^1, \sigma^2)$ therefore $\mathcal{U}(\sigma_A^1, \sigma^2) = 0$. □

We now show that in any Nash equilibrium, the defender places more probability on selecting more expensive sets of nodes (Note: it is true only inside the attacker's support). The intuition to this lemma is clear; Out of the attacker's support, if the defender will tend to choose the more costly groups of nodes, then he "forces" the attacker to pay more for the attack.

Lemma 3.2. *Let (σ^1, σ^2) be a Nash equilibrium. Then $\forall B_1, B_2 \in supp(\sigma^2)$ it holds that $C(B_1) \leq C(B_2)$ if and only if*

$$\sum_{A \in S^1 s.t. A \subseteq B_1} \sigma_A^1 \leq \sum_{A \in S^1 s.t. A \subseteq B_2} \sigma_A^1$$

Proof. Reminder: If v is the value of a general zero-sum game G and (σ^1, σ^2) is an equilibrium, then any pure strategies from the support $A \in supp(\sigma^1), B \in supp(\sigma^2)$ can be used to achieve the value of the game, i.e. $G(A, \sigma^2) = G(\sigma^1, B) = v$.

Now, turning back to the proof, if $\sigma^2_B \neq 0$ (for some $B \subset V$), then $\mathcal{U}\left(\sigma^1, S^2_B\right) = v$. Therefore:

$$v = \mathcal{U}\left(\sigma^1, S^2_{B_1}\right) = C(B_1) - W_{att} \cdot \sum_{A \in S^1 s.t. A \subseteq B_1} \sigma^1_A$$

$$v = \mathcal{U}\left(\sigma^1, S^2_{B_2}\right) = C(B_2) - W_{att} \cdot \sum_{A \in S^1 s.t. A \subseteq B_2} \sigma^1_A$$

The first step in each row is from $B_1, B_2 \in supp(\sigma^2)$, the second is from the definition of $\mathcal{U}$. Therefore

$$C(B_1) - C(B_2) = W_{att} \cdot \left(\sum_{A \in S^1 s.t. A \subseteq B_1} \sigma^1_A - \sum_{A \in S^1 s.t. A \subseteq B_2} \sigma^1_A \right)$$

Thus,

$$C(B_1) \leq C(B_2) \leftrightarrow \sum_{A \in S^1 s.t. A \subseteq B_1} \sigma^1_A \leq \sum_{A \in S^1 s.t. A \subseteq B_2} \sigma^1_A$$

□

An immediate corollary that will later prove useful is that corruptable subsets in the attacker's support with the same cost are equally likely to contain the defender's choice of connections.

Corollary 3.3. *The probability of choosing a group of peers is determined by the attacker's support. I.e., let (σ^1, σ^2) be an equilibrium, then $\forall B_1, B_2 \in supp(\sigma^2)$, it holds that $C(B_1) = C(B_2)$ if and only if*

$$\sum_{U \subseteq B_1 \cap S^1} \sigma^1_U = \sum_{U \subseteq B_2 \cap S^1} \sigma^1_U$$

4 BUFFER-RESTRICTED IMPLEMENTATION

We now turn to the scenario where buffers are restricted. Our goal here is to decrease the number of addresses that we should remember in the buffer, while obtaining the same game value. This section aims to match a more realistic assumption, in which the nodes can not store all the peers in the network (due to very high number of nodes or low memory capacity).

Intuitively, we want our buffer-management algorithm to follow Nash equilibrium strategies, which will also provide a safety level. We will do so by defining a new game, $\hat{\mathcal{U}}$, and a modified strategy space for the defender using equivalence classes on the original strategies S^1. We then prove that this new game has the same value. The main result of this section is Theorem 3, in which we show that we can implement a Nash equilibrium strategy using only a buffer of size $O(|\hat{S^1}|)$. Finally, in the next section, we will show that this number is small enough in practice.

Example 4.1 (Toy Example & Intuition). Assume that the set of nodes $|V|$ is large, and that there are only two types of nodes in V: blue and red. Assume that the number of connections is $H = 1$, and that the defender's Nash equilibrium strategy is to choose to connect to a red node with probability $\frac{2}{3}$, and to some blue node with probability $\frac{1}{3}$ (choosing uniformly between nodes with the same color). A simple buffer management algorithm will do the following: store one blue node and one red. Make sure that the nodes that we stored were chosen uniformly from the set of nodes with the same color. Then, from the nodes in the buffer, select red with probability $\frac{2}{3}$ or blue with probability $\frac{1}{3}$. In this way, the equilibrium probability is induced using limited buffer space, regardless of the original size of V.

We begin by providing a definition of equivalence classes (EC) of strategies. Intuitively, we say that two strategies are considered equivalent if they are selected with equal probability in any Nash equilibrium. This will enable us to store few representative addresses from each such EC.

Equivalent strategies Define the equivalence relation $A_1 \sim A_2$ on sets of nodes $A_1, A_2 \in S^1$ if $\forall(\sigma^1, \sigma^2)$ Nash equilibrium in $\mathcal{U}$, holds that $\sigma^1_{A_1} = \sigma^1_{A_2}$. We will denote this EC with $[A_1]$, and its size with $|[A_1]|$.

We now define a new game, $\hat{\mathcal{U}}$, that uses these ECes as the defender's strategies space, and mark it with $\hat{S^1}$. In this new game, the defender does not distinguish between different connections from the same EC (the game is thus equivalent to choosing uniformly from the class and playing that strategy in the original game $\mathcal{U}$).

The new game is formally defined as follows: Let $[A] \in \hat{S^1}$ be defender's strategy in $\hat{\mathcal{U}}$, and $U \in S^2$ be an attacker strategy. The new game's utility:

$$\hat{\mathcal{U}}([A], U) = \frac{\sum_{A_1 \in [A]} \mathcal{U}(A_1, U)}{|[A]|} = C(U) - W_{att} \cdot \frac{\sum_{A_1 \in [A]} \delta_{A_1 \subset U}}{|[A]|}$$

The following theorem discusses the analogy between these two games. We show that we can "map" Nash equilibrium strategies from one game to the other. We will use this theorem later, when we will explore Nash equilibria in $\hat{\mathcal{U}}$, and deduce the correlated strategies in $\mathcal{U}$.

Theorem 2. *Nash equilibria in both games have the same game value. Moreover, there exists a function $T : \Delta_{S^1} \to \Delta_{\hat{S^1}}$ such that (σ^1, σ^2) is a Nash equilibrium in $\mathcal{U}$ iff $(T(\sigma^1), \sigma^2)$ is a Nash equilibrium in $\hat{\mathcal{U}}$.*

To prove the theorem we formally define a mapping that "translates" between the two games:

Take $T : \Delta_{S^1} \to \Delta_{\hat{S^1}}$ by $\left(T(\sigma)\right)_{[A]} = \sum_{B \in [A]} \sigma_B$ and the inverse $T^{-1} : \Delta_{\hat{S^1}} \to \Delta_{S^1}$ by $\left(T^{-1}(\hat{\sigma})\right)_A = \frac{\hat{\sigma}_{[A]}}{|[A]|}$

The following lemma gives us properties that T holds. We will use this properties in the proof of the theorem.

Lemma 4.2. *For all $\hat{\sigma^1} \in \Delta_{\hat{S^1}}, \sigma^2 \in \Delta_{S^2}$ it holds that: (i) T^{-1} does not change the utility, i.e. $\hat{\mathcal{U}}(\hat{\sigma^1}, \sigma^2) = \mathcal{U}(T^{-1}(\hat{\sigma^1}), \sigma^2)$ (ii) $T(T^{-1})$ is the identity, i.e. $T(T^{-1}(\hat{\sigma^1})) = \hat{\sigma^1}$. In addition, if (σ^1, σ^2) is Nash equilibrium in $\mathcal{U}$, then (iii) $T^{-1}(T(\sigma^1)) = \sigma^1$ and (iv) T does not change the utility. i.e. $\hat{\mathcal{U}}(T(\sigma^1), \sigma^2) = \mathcal{U}(\sigma^1, \sigma^2)$*

Proof. First note that:

$$\sum_{B \subset V} \sum_{[A] \in \hat{S^1}} \hat{\sigma^1}_{[A]} \sigma^2_B \frac{\sum_{A_1 \in [A]} \delta_{A_1 \subset B}}{|[A]|} =$$

$$\sum_{B \subset V} \sum_{[A] \in \hat{S^1}} \sum_{A_1 \in [A]} \hat{\sigma^1}_{[A]} \sigma^2_B \frac{\delta_{A_1 \subset B}}{|[A]|} =$$

$$\sum_{B\subset V}\sum_{[A]\in\hat{S^1}}\sum_{A_1\in[A]}\left(T^{-1}(\hat{\sigma^1})_{A_1}\cdot|[A]|\right)\sigma_B^2\frac{\delta_{A_1\subset B}}{|[A]|}=$$
$$\sum_{B\subset V}\sum_{[A]\in\hat{S^1}}\sum_{A_1\in[A]}T^{-1}(\hat{\sigma^1})_{A_1}\sigma_B^2\delta_{A_1\subset B}=$$
$$\sum_{B\subset V}\sum_{A_1\in S^1}T^{-1}(\hat{\sigma^1})_{A_1}\sigma_B^2\delta_{A_1\subset B}$$

Where the second step is $T^{-1}(\hat{\sigma^1})_{A_1}=\frac{\hat{\sigma^1}_{[A]}}{|[A]|}$, and the last step is because $S^1=\bigcup_{[A]\in\hat{S^1}}\{A_1|A_1\in[A]\}$. Then we get:

$$\hat{\mathcal{U}}(\hat{\sigma^1},\sigma^2)=\sum_{B\subset V}\sigma_B^2C(B)-W_{att}\sum_B\sum_{[A]}\hat{\sigma^1}_{[A]}\sigma_B^2\frac{\sum_{A_1\in[A]}\delta_{A_1\subset B}}{|[A]|}=$$
$$\sum_{B\subset V}\sigma_B^2C(B)-W_{att}\sum_B\sum_{A_1\in S^1}T^{-1}(\hat{\sigma^1})_{A_1}\sigma_B^2\delta_{A_1\subset B}=$$
$$\mathcal{U}(T^{-1}(\hat{\sigma^1}),\sigma^2)$$

Which proves (i) from the lemma.

To prove (ii) we use directly the definitions. we will show that for every $[A]\in\hat{S^1}$ it holds that $T(T^{-1}(\hat{\sigma^1}))_{[A]}=\hat{\sigma^1}_{[A]}$:

$$T(T^{-1}(\hat{\sigma^1}))_{[A]}=\sum_{B\in[A]}T^{-1}(\hat{\sigma^1})_B=\sum_{B\in[A]}\frac{\hat{\sigma^1}_{[A]}}{|[A]|}=|[A]|\cdot\frac{\hat{\sigma^1}_{[A]}}{|[A]|}$$

Where the first step is the definition of T and the second is the definition of T^{-1}.

Proving (iii) will again, use the definition directly: if (σ^1,σ^2) is Nash equilibrium, then for any $[A]\in\hat{S^1}$, any $A_1,A_2\in[A]$ holds $\sigma^1_{A_1}=\sigma^1_{A_2}$ (from the definition of the EC). Therefore we get $T(\sigma^1)_{[A]}=\sum_{B\in[A]}\sigma^1_B=|[A]|\cdot\sigma^1_{A_1}$ (the first step is the definition of T). And finally, $T^{-1}(T(\sigma^1))_{A_1}=\frac{T(\sigma^1)_{[A]}}{|[A]|}=\frac{|[A]|\cdot\sigma^1_{A_1}}{|[A]|}=\sigma^1_{A_1}$ where the first step is the definition of T^{-1}.

To finish and prove (iv): if (σ^1,σ^2) is Nash equilibrium in $\mathcal{U}$, then $\mathcal{U}(\sigma^1,\sigma^2)=\mathcal{U}(T^{-1}(T(\sigma^1)),\sigma^2)=\hat{\mathcal{U}}(T(\sigma^1),\sigma^2)$, where the first step is (iii) and the second is (i). □

Now, after we have the properties of the mapping between the two games, we can dive into the proof of Theorem 2:

Proof of Theorem 2. We now show that the function T preserves any Nash equilibrium in the translation.

Let (σ^1,σ^2) be a Nash equilibrium in $\mathcal{U}$, we want to prove that $(T(\sigma^1),\sigma^2)$ is Nash equilibrium in $\hat{\mathcal{U}}$. On the one hand, $\forall\sigma'^1\in\hat{S^1}$, we need to show that $\hat{\mathcal{U}}(T(\sigma^1),\sigma^2)\geq\hat{\mathcal{U}}(\sigma'^1,\sigma^2)$.

Indeed, it holds

$$\hat{\mathcal{U}}(\sigma'^1,\sigma^2)=\mathcal{U}(T^{-1}(\sigma'^1),\sigma^2)\leq\mathcal{U}(\sigma^1,\sigma^2)=\hat{\mathcal{U}}(T(\sigma^1),\sigma^2)$$

where the first equality is (i) from Lemma 4.2, the second is the definition of Nash equilibrium in $\mathcal{U}$, and the last is (iv).

On the other hand, $\forall\sigma'^2\in S^2$

$$\hat{\mathcal{U}}(T(\sigma^1),\sigma'^2)=\mathcal{U}(\sigma^1,\sigma'^2)\geq\mathcal{U}(\sigma^1,\sigma^2)=\hat{\mathcal{U}}(T(\sigma^1),\sigma^2)$$

where the first and last equality are (iv), and the second is the definition of Nash equilibrium.

Thus we have shown that this is an equilibrium. They have the same value due to (iv): $\mathcal{U}(\sigma^1,\sigma^2)=\hat{\mathcal{U}}(T(\sigma^1),\sigma^2)$

The other direction (if $(\hat{\sigma^1},\sigma^2)$ is a Nash equilibrium in $\hat{\mathcal{U}}$, then $(T^{-1}(\sigma^1),\sigma^2)$ is Nash equilibrium in $\mathcal{U}$) could be proven in the same way. □

From Lemma 4.2 we get that, in addition to Nash equilibria, T also preserves safety levels in the games:

Corollary 4.3. *Defender's safety level l in $\hat{\mathcal{U}}$ can be translated to a safety level $\geq l$ in $\mathcal{U}$.*

Proof. Let $\hat{\sigma}^1\in\hat{S^1}$ be a safety level l in $\hat{\mathcal{U}}$. Indeed, $\forall\sigma^2\in S^2$, $\mathcal{U}(T^{-1}(\hat{\sigma^1}),\sigma^2)=\hat{\mathcal{U}}(\hat{\sigma^1},\sigma^2)\geq l$ where the equality on the left is (i), and the inequality on the right is the definition of safety level l in $\hat{\mathcal{U}}$. Thus $T^{-1}(\hat{\sigma^1})$ is safety level l in $\mathcal{U}$ □

Equivalent peers Thus far, we have considered equivalent strategies. We define a similar equivalence relation on the nodes in V: Two nodes u,v are equivalent if adding them to another group of nodes is strategically equivalent. This definition is slightly subtle (nodes don't always have the same cost for attackers – this depends on the set of other nodes that are chosen).

Formally, we denote: $u\rightleftharpoons v$, if for every $A\subset V,|A|=H-1$ it holds that $A\cup\{u\}\sim A\cup\{v\}$.

Let $[v]_{\rightleftharpoons}$ denote the equivalence class (EC) of node $v\in V$, and the ECes space with $\hat{V}:=\{[v]_{\rightleftharpoons}|v\in V\}$.

Note that $[\{v_1,\cdots,v_H\}]\neq[v_1]_{\rightleftharpoons}\times\cdots\times[v_H]_{\rightleftharpoons}$ and generally there is no containment in either direction (see Corollary 4.5 for a specific interesting property).

4.1 The Buffer Management Algorithm

In this section we propose a concrete and practical way of implementing the buffer management algorithm $\mathscr{R}$ (as described in Subsection 2.3), that utilizes a Bloom filter. The main problem that this section overcomes is the problem of an attacker that re-propagates the same nodes again and again. In this scenario, the attacker assumes that the user "forgets" nodes that are not in his buffer. Therefore the attacker can propagate his nodes repeatedly, and ensuring that they will be entered to the defender's buffer. The initial discussion will assume a single choice of connections, and Subsection 4.1.1 will generalize it to the continuous game.

A Bloom filter [1] is a data structure that uses hash-coded information to encode a set of items. It allows for some small fraction of errors in membership tests (false positives). The error rate can be lowered by increasing memory usage. In our paper, we use this method to avoid addresses that we already saw including those evicted from the buffer. We use the Bloom filter's deterministic answer to completely avoid known addresses, and accept a small fraction of false-positive answers on newly received addresses.

For a set of addresses $X\subseteq V$ and a strategy $A\in S^1$, define $X|_{[A]}\subseteq S^1$ to be all the strategies inside the equivalence class $[A]$ that can be played using addresses in the set X. I.e. a strategy (which is a set of addresses) B is in $X|_{[A]}$ iff $B\in[A]$ and $B\subseteq X$.

The next theorem discusses the benefits of well-implemented buffer management algorithms, and the following lemma gives a possible implementation:

Theorem 3. *Let I be the set of all the addresses sent to the defender, and let $\mathbb{B}$ be the set of all addresses in the buffer. Assume that we have*

some buffer management algorithm $\mathscr{R}$ that fills up the buffer and then evicts addresses for every new insertion. Assume also that for a given equivalence class of strategies $[A] \in \hat{S^1}$, uniform selection of strategy in $\mathbb{B}|_{[A]}$ is equivalent to uniform selection in $I|_{[A]}$.

Then we can implement any Nash strategy or safety level on a restricted buffer of size $O(|\hat{S^1}|)$ with the same value as the game on a limitless buffer.

Proof. For any Nash Equilibrium strategy or safety level in $\mathcal{U}$, translate it to the correlate strategy in $\hat{\mathcal{U}}$ (Using Theorem 2 / Corollary 4.3), and get a strategy $[A] \in \hat{S^1}$ that we should play. This means that we should choose uniformly a set of connections from $I|_{[A]}$, and by the theorem's assumptions, it is equivalent to choose uniformly from $\mathbb{B}|_{[A]}$.

Therefore, we implement the strategy over a restricted buffer with the same game value of $\mathcal{U}$. □

Algorithm 1 describes the buffer management policy that we propose. Intuitively, the buffer is partitioned into buckets that correspond to different equivalence classes. When an address is sent to the node, it is mapped to a specific bucket. If it is the i'th address to appear for that bucket (since the last reset of the Bloom filter) it is accepted with probability $\min(1, bucket\_size/i)$, and a uniformly selected address is evicted from the bucket if it is full. This behavior gives us the properties we desire:

Lemma 4.4. *Algorithm 1 with $EG = \hat{S^1}$ and an optimal filter*[1] *satisfies the conditions of Theorem 3.*

Proof. For any equivalence class in the defender's strategy space ("bucket"), we should prove the following claim: If we choose a node uniformly from the bucket that we stored in the buffer, then it has the same probability as choosing it uniformly from all the nodes that the defender is notified about and belong to this equivalence class (i.e., the case where the bucket size is infinite). Assume that we make the choice after being notified of l nodes in this bucket. We need to prove that there is a chance of $\frac{1}{l}$ to choose any node. Indeed, $\forall j \in \{1, \cdots, l\}$, we insert it into the buffer with probability $\frac{bucket\_size}{j}$ and it will still be there after the next input with probability $(1 - \frac{bucket\_size}{j+1} \frac{1}{bucket\_size}) = \frac{j}{j+1}$, so after l input nodes: $\frac{j}{j+1} \frac{j+1}{j+2} \cdots \frac{l-1}{l} = \frac{j}{l}$, therefore the total probability of the j'th node to be in the bucket after a total of l inputs is: $\frac{bucket\_size}{j} \cdot \frac{j}{l} = \frac{bucket\_size}{l}$. Finally, the probability of choosing any node from the bucket is $\frac{bucket\_size}{l} \cdot \frac{1}{bucket\_size} = \frac{1}{l}$ (we choose uniformly in the bucket), which is the desired property. □

The next corollary shows a different perspective, wherein we implement the Nash equilibrium using the equivalent classes in $\hat{V}$ (instead of $\hat{S^1}$). This is possible only in the specific case that the equivalence classes of strategies are built from equivalence classes of nodes.

Corollary 4.5. *Denote all the addresses received by the defender as the set I, and all the addresses in the buffer with $\mathbb{B}$. If for every $v_1, \cdots, v_n \in V$ it holds that $[v_1]_{\rightleftharpoons} \times \cdots \times [v_H]_{\rightleftharpoons} \subseteq [v_1, \cdots, v_n]$, then we can implement any Nash strategy or safety level on a restricted buffer of size $O(H \cdot |\hat{V}|)$ with the same value as the limitless buffer game.*

[1] A filter without false positive or false negative—essentially a very large Bloom filter.

Input: *EG* - the buckets, $\mathcal{B}$ - size of the Bloom filter, *bucket_size*
BF := Bloom-filter buffer of size $\mathcal{B}$;
∀ bucket ∈ EG: bucket_history[bucket]=0 ;
∀ bucket ∈ EG: buckets[bucket]=∅ ;
while *n := new input node* **do**
  if *not BF.contains(n)* **then**
    b := n.bucket ;
    bucket_history[b] ++ ;
    BF.add(n) ;
    begin With Pr. $\frac{bucket\_size}{bucket\_history[b]}$
      if *buckets[b].isFull* **then**
        buckets[b].uniformlyRemoveOne ;
      buckets[b].add(n) ;

Algorithm 1: $\mathscr{R}$ algorithm for equivalence classes

Proof. We use Algorithm 1 with $EG = \hat{V}$ and $bucket\_size = H$, i.e. store H nodes for any equivalence class in $\hat{V}$. Easy modifications to the previous proof show that for a given $[v]_{\rightleftharpoons} \in \hat{V}$, choosing uniformly from $I \cap [v]_{\rightleftharpoons}$ is equivalent to choose uniformly from $\mathbb{B} \cap [v]_{\rightleftharpoons}$.

Now, for every Nash equilibrium strategy $A = \{v_1, \cdots, v_H\} \in S^1$, we can choose nodes uniformly from $\mathbb{B} \cap [v_1]_{\rightleftharpoons}, \cdots, \mathbb{B} \cap [v_n]_{\rightleftharpoons}$, mark the group of the chosen nodes with $\hat{A}$. Then according to the assumption of this corollary, $\hat{A} \in [A]$, and therefore playing $\hat{A}$ gives the same game value as A. □

4.1.1 Continuous Games: Refreshing the Buffer and Bloom Filter. The above algorithm works for a single "round", where we are a node in the network, establishing the known nodes list, and choosing connections for the next round. This is not enough for a continuous game, where nodes in the network come and go frequently, and we need to choose connections repeatedly, using the same buffer. Moreover, over populating the bloom filter increases the probability of false-positives.

To overcome these difficulties, we can direct the network protocol to propagate live nodes every $\mathcal{T}$ time units, and store two copies of the buffer and the bloom filter that reset alternately every $2\mathcal{T}$ time units. This method gives us the ability to remember IPs from a window of $\mathcal{T}$, which is the full available information on the network (as we've assumed honest nodes send their IP address to others at least once every $\mathcal{T}$).

Additionally, in more realistic scenarios in which churn is an issue, and honest nodes may be offline at times, we suggest using larger bucket sizes to preserve a sufficiently large set of alternative connections.

The next lemma shows that even if some IPs in the bucket cannot be selected, e.g., if they are stale, selecting uniformly from the remaining nodes in the bucket is equivalent again to a uniform selection (from the whole set of addresses). This will show that Algorithm 1 satisfies the conditions to Theorem 3 also in the scenario of churn.

Lemma 4.6 (Lemma 4.4 for continuous games). *Let I be the set of all the addresses sent to the defender, and let $\mathbb{B}$ be the set of all addresses in the buffer which is managed using Algorithm 1 with $bucket\_size > 1$. Denote the defender's chosen strategy with $[A] \in \hat{S}^1$, and a strategy $B \in [A]$ which is churned (and can not be played). Then, choose uniformly from $\mathbb{B}|_{[A]} - \{B\}$ is equivalent to choose uniformly from the entire input without B, i.e. from $(I - \{B\})|_{[A]}$.*

The proof is equivalent to that of Lemma 4.4, when splitting to two cases whether B entered the buffer or not.

5 INSTANTIATING THE COST FUNCTION

In this section we choose a concrete cost function for the cost of corrupting IPs. We begin with a trivial example that shows that the constant cost function in our model yields the expected uniform selection over all IPs.

Example 5.1. Using the notation of Subsection 2.1.1, let $C(U) = c \cdot |U|$ (for some $c \in \mathbb{R}_+$). Therefore, if the defender will choose the subset of nodes A and the attacker corrupts B, then the utility of the game is $c \cdot |B| - W_{att} \cdot \delta_{A \subset B}$.

Due to Lemma 3.2 and the symmetry of the nodes in this example, the defender will choose any set of nodes with *the same probability* in *every* Nash equilibrium.

In this case, it is easy to see that there is a single defender's equivalence class, i.e. $|\hat{S^1}| = 1$. Therefore, by Theorem 3, we know that we can implement this on a restricted buffer and get the same game value.

I.e. using Algorithm 1 with $|EG| = 1$, $bucket\_size = H$ indeed works: it returns H nodes that are uniformly distributed over all the input (Lemma 4.4).

5.1 A Cost Function Based on IP Masks

We now focus on a specific game where the cost function is determined by the IP/16 prefix (or "mask"), i.e. acquiring an IP range costs c_{new} and every IP from that range costs c_{node}. The intuition behind the use of this cost function comes from the way IPs are allocated and routed on the internet. Packets are forwarded usually according to the prefixes of the IP addresses, and IPs with a similar prefix are usually hosted by the same provider. As a further justification, we examined a variety of cloud service providers and found that in practice, obtaining two IP addresses with the same /16 prefix is about 4-5 times cheaper than obtaining two IP addresses with different prefixes (Prefixbroker, AWS, Azure).

We thus set the cost of obtaining a set of IPs A as:

$$C(A) = c_{new} \cdot number\_of\_masks\_in\_A + c_{node} \cdot |A|$$

Our main practical result is a direct consequence of the theoretical analysis in the previous section:

Theorem 4 (Theorem 3 for ip mask costs). *We can implement the game with the cost function of IP/16 prefixes on a limited buffer, and get the game value as the unrestricted-buffer game.*

Proof. According to Corollary 3.3, we know that if v_1, v_2 are nodes with the same /16 prefix then $[v_1]_{\rightleftharpoons} = [v_2]_{\rightleftharpoons}$ (because we should always choose them with the same probability).

Therefore, any defender's strategy $A = \{v_1, \cdots, v_H\} \in S^1$, holds the property that: $[v_1]_{\rightleftharpoons} \times \cdots \times [v_H]_{\rightleftharpoons} \subseteq [A]$.

Finally, our theorem follows from Corollary 4.5. □

Size of the Buffer Practically, there are 2^{16} possibilities for /16 prefixes. Therefore if every record contains an IP address and timestamp (total of $2^6 = 64$ bits), the buffer's overall space will be $2^{22} bit = 524KB$.

6 EVALUATION

In the following section, we evaluate the results with respect to the IP-masks based cost function. We focus on a game in which the number of nodes present in each subnet is *not* identical. This is a very realistic setting, but it also makes the Nash equilibrium harder to compute. Due to this difficulty, we will define a relaxed game, $\bar{\mathcal{U}}$, wherein we limit the defender by reducing his available strategies. Those restrictions give us only a safety level for the original game; but one that we can easily compute and apply.

First, note that the defender should not distinguish between two /16 prefixes that have the same number of nodes (in any equilibrium, prefixes with the same number of nodes are selected with the same probability). Therefore, the strategic equivalence classes are defined by the number of nodes inside each prefix.

We introduce some notations: Let M_a be the number of all the nodes in all the prefixes with size a. Let $avg_a = \frac{c_{new} + a \cdot c_{node}}{a}$ denote the average cost for node in prefix of size a. Finally, $x_a \in \mathbb{N}$ is the number of nodes that the attacker corrupts in a prefix of size a. The attacker invests at least $x_a \cdot avg_a$, and the fraction of corrupted nodes is $\frac{x_a}{M_a}$.

Now, we relax the game by limiting the defender in two ways: (i) He cannot distinguish between two prefixes with the same size, and (ii) he must choose connections independently, each with probability p_a.

The defender chooses whether he would like to take addresses from prefixes of size a. Let $y_a \sim Bin(H, p_a)$ denote this choice. That is, $y_a \in \mathbb{N}$ is the number of connections that the defender chooses from prefix-size a, and 0 otherwise. Then

$$\bar{\mathcal{U}}(\bar{x}, \bar{y}) \geq \sum_a x_a \cdot avg_a - W_{att} \cdot \prod_a \left(\frac{x_a}{M_a}\right)^{y_a}$$

I.e., the game value increases linearly with large constants but decreases exponentially with small constants.

Simple Evaluation of $\bar{\mathcal{U}}$ To retrieve an actual value for the calculation above, we examined the behavior of the Bitcoin network. We collected (using the site blockchain.com) a snapshot of all the nodes that connected to the network. We then deduced the number of different /16 prefixes in which nodes were located. The distribution of the nodes in the different prefixes is shown in Figures 1.

We calculated the sizes of each mask size and got: $\prod_a \frac{1}{M_a} \approx 8.991 \cdot 10^{-52}$. Therefore the attacker needs to invest huge amounts of resources to gain a negative value.

6.1 Comparing to a Naive Benchmark

In this section, we compare our results to two naive benchmarks, where each node chooses its connections uniformly from the buffer, and the buffer is chosen uniformly from the whole network. The first benchmark (the more naive) assumes that the node may be

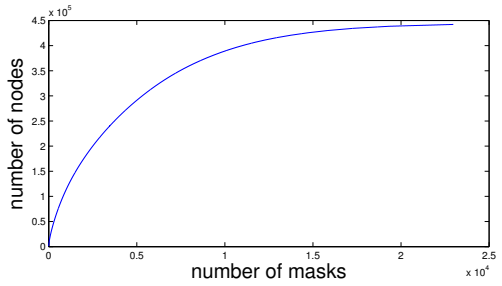

Figure 1: Number of IP/16 prefixes in the Bitcoin network (most populated masks first)

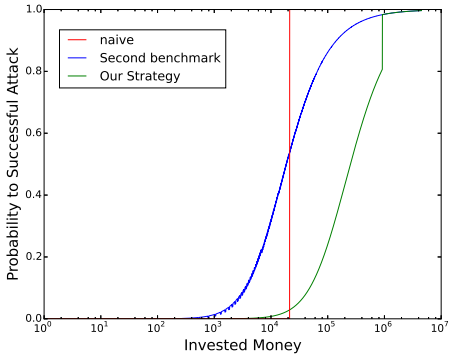

Figure 2: Probability of a successful attack by investment.

subject to repeated transmissions of the same IP, which it cannot detect unless that IP is already in the buffer. The second benchmark adds an accompanying Bloom filter to filter out re-transmissions. Considering our method, an attacker's best response is to create nodes in such a way that p_a is identical for any mask a. Then, we calculate the probability of being successfully attacked as a function of the attacker's investment. We consider a network and configuration respectively to Bitcoin's network topology, buffer size (= 20480), number of connections ($H = 8$). We choose $\frac{c_{new}}{c_{node}} = 10$ using the approximated prices on Amazon EC2, which was sampled on June 4, 2020. The results are shown[2] in Figure 2. Note that the attacker's cost is on a logarithmic scale.

6.2 Isolate Entire Subset of Nodes

In this subsection, we broaden our focus and evaluate the effectiveness of this approach to prevent the isolation of an entire subset of nodes.

The above analysis introduces the game between the attacker and a single defender, and deduced strategies to each selfish honest node. The following evaluation will show that, in practice, this selfish behavior also increases the cost of an attack to isolate an entire set of nodes. Therefore, this is an effective strategy in a network perspective as well.

[2]The "spike" in the graph is caused by the number of existing masks (if the attacker corrupts in all masks). In this case, the probability of a successful attack is the same as in the naive approach.

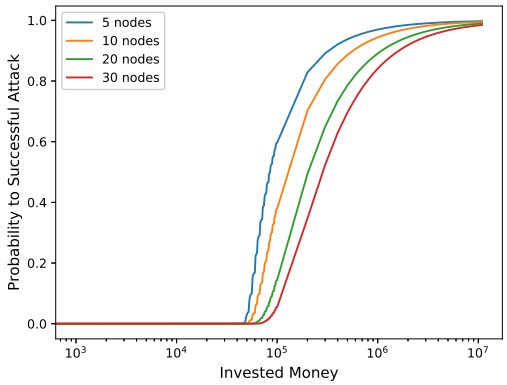

Figure 3: Probability of a successful isolate entire subset of nodes.

In this scenario, we relax the demand that each node will connect to at least one honest node (because the attacker "allows" connection between the nodes inside the isolated group). On the other hand, the attack will fail if at least one of the isolated nodes will connect to an honest node outside the isolated group.

Figure 3 shows the probability of successful attack as a function of the invested money, and for different sizes of isolated groups. We can see that when the attacker invests around 10^5\$ (USD), the probability to a successful attack is much lower for bigger groups. Nevertheless, as we assumed in the Section 2, an attack from unrestricted attacker is inevitable.

7 FUTURE WORK & CONCLUSIONS

In this work we explored a game theoretic model for blockchain systems' P2P network formation. Our results indicate that by using a model for the costs of the attacker, it is possible to select each node's connections so as to reduce the likelihood of being isolated. We presented an example strategy that can be implemented in any blockchain system with overlay P2P network (such as Bitcoin, Ethereum, etc.). We then examined the cost of a successful attack in our model, and compared the results to other strategies.

Future work should extend the model to games that are not zero-sum, and try to better account for attackers attempting to isolate much larger chunks of the network. We assume that an attacker tries to isolate a specific subset of nodes, and that the attack fails if one of the nodes has a connection to an honest non-attacked node. This of course can be generalized to an attacker that tries to isolate "as many as possible" out of specific nodes or the whole network. Moreover, it is interesting to examine the resulting topology (properties such as diameter and centrality) when evaluating each strategy.

This paper uses a single, existing network characteristic for the cost function, the IP mask, to ease the implementation on current blockchain systems. This is not mandatory, and there is more information about peers that we can utilize such as round-trip time, traceroute, and other network features. Another possibility is to

augment nodes with additional information that will be hard for an attacker to fake. For example, it may be interesting to examine mechanisms of proof-of-work that might increases the reliability of a node in the network.

Finally, for the most general game form we used a specific strategy and compared it to other benchmarks. Finding a way to solve the game without assumptions or heuristic strategies, or finding a defender's strategy that is in some Nash equilibrium, will yield the optimal algorithm for any mask-based blockchain system.

8 ACKNOWLEDGEMENTS

This research was supported by the Israel Science Foundation (grant 1504/17) and by a grant from the HUJI Cyber Security Research Center in conjunction with the Israel National Cyber Bureau.

REFERENCES

[1] Burton H Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* 13, 7 (1970), 422–426.
[2] Mauro Conti, E Sandeep Kumar, Chhagan Lal, and Sushmita Ruj. 2018. A survey on security and privacy issues of bitcoin. *IEEE Communications Surveys & Tutorials* 20, 4 (2018), 3416–3452.
[3] Maya Dotan, Yvonne-Anne Pignolet, Stefan Schmid, Saar Tochner, and Aviv Zohar. 2020. Survey on Cryptocurrency Networking: Context, State-of-the-Art, Challenges. *arXiv preprint arXiv:2008.08412* (2020).
[4] John R Douceur. 2002. The sybil attack. In *Peer-to-peer Systems*. Springer, 251–260.
[5] Ittay Eyal. 2015. The miner's dilemma. In *2015 IEEE Symposium on Security and Privacy*. IEEE, 89–103.
[6] Ittay Eyal and Emin Gün Sirer. 2018. Majority is not enough: Bitcoin mining is vulnerable. *Commun. ACM* 61, 7 (2018), 95–102.
[7] Andrew Fielder, Emmanouil Panaousis, Pasquale Malacaria, Chris Hankin, and Fabrizio Smeraldi. 2014. Game theory meets information security management. In *IFIP International Information Security Conference*. Springer, 15–29.
[8] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. 2015. Eclipse attacks on bitcoin's peer-to-peer network. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*. 129–144.
[9] Mojtaba Jamshidi, Ehsan Zangeneh, Mehdi Esnaashari, Aso Mohammad Darwesh, and Mohammad Reza Meybodi. 2019. A novel model of sybil attack in cluster-based wireless sensor networks and propose a distributed algorithm to defend it. *Wireless Personal Communications* 105, 1 (2019), 145–173.
[10] Hongling Jiang and Xiuli Shao. 2014. Detecting P2P botnets by discovering flow dependency in C&C traffic. *P2P Networking and Applications* 7, 4 (2014), 320–331.
[11] Hyun Mi Jung, Il-Sun Hwang, Jeong-Kyung Moon, and Hark-Soo Park. 2016. A security monitoring method for malicious P2P event detection. *Peer-to-Peer Networking and Applications* 9, 3 (2016), 498–507.
[12] Iuon-Chang Lin and Tzu-Chun Liao. 2017. A survey of blockchain security issues and challenges. *IJ Network Security* 19, 5 (2017), 653–659.
[13] Mohammad Hossein Manshaei, Quanyan Zhu, Tansu Alpcan, Tamer Bacşar, and Jean-Pierre Hubaux. 2013. Game theory meets network security and privacy. *ACM Computing Surveys (CSUR)* 45, 3 (2013), 25.
[14] Yuval Marcus, Ethan Heilman, and Sharon Goldberg. 2018. Low-Resource Eclipse Attacks on Ethereum's Peer-to-Peer Network. *IACR Cryptology ePrint Archive* 2018 (2018), 236.
[15] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. *Consulted* 1, 2012 (2008).
[16] James Newsome, Elaine Shi, Dawn Song, and Adrian Perrig. 2004. The sybil attack in sensor networks: analysis & defenses. In *Third international symposium on information processing in sensor networks, 2004. IPSN 2004.* IEEE, 259–268.
[17] Carlos Pinzón and Camilo Rocha. 2016. Double-spend attack models with time advantange for bitcoin. *Electronic Notes in Theoretical Computer Science* 329 (2016), 79–103.
[18] James Pita, Manish Jain, Janusz Marecki, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. 2008. Deployed ARMOR protection: the application of a game theoretic model for security at the Los Angeles International Airport. In *Proceedings of AAMAS*. International Foundation for Autonomous Agents and Multiagent Systems, 125–132.
[19] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. 2016. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security*. Springer, 515–532.
[20] Atul Singh et al. 2006. Eclipse attacks on overlay networks: Threats and defenses. In *In IEEE INFOCOM*. Citeseer.
[21] Muoi Tran, Inho Choi, Gi Jun Moon, Anh V Vu, and Min Suk Kang. 2020. A Stealthier Partitioning Attack against Bitcoin Peer-to-Peer Network. In *2020 IEEE Symposium on Security and Privacy (SP)*. 496–511.
[22] Guido Urdaneta, Guillaume Pierre, and Maarten Van Steen. 2011. A survey of DHT security techniques. *ACM Computing Surveys (CSUR)* 43, 2 (2011), 8.
[23] Gavin Wood et al. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151, 2014 (2014), 1–32.
[24] Karl Wüst and Arthur Gervais. 2016. *Ethereum eclipse attacks*. Technical Report. ETH Zurich.
[25] HAN Xinhui, XIAO Xianquan, Jianyu ZHANG, Bingshuang LIU, and Yuan ZHANG. 2015. Sybil defenses in DHT networks based on social relationships. *Journal of Tsinghua University (Science and Technology)* 54, 1 (2015), 1–7.
[26] Haifeng Yu, Michael Kaminsky, Phillip B Gibbons, and Abraham Flaxman. 2006. Sybilguard: defending against sybil attacks via social networks. *ACM SIGCOMM Computer Communication Review* 36, 4 (2006), 267–278.
[27] Adja Elloh Yves-Christian, Badis Hammi, Ahmed Serhrouchni, and Houda Labiod. 2018. Total Eclipse: How To Completely Isolate a Bitcoin Peer. In *2018 Third International Conference on Security of Smart Cities, Industrial Control System and Communications (SSIC)*. IEEE, 1–7.
[28] Alireza Zarreh, Can Saygin, HungDa Wan, Yooneun Lee, and Alejandro Bracho. 2018. A game theory based cybersecurity assessment model for advanced manufacturing systems. *Procedia Manufacturing* 26 (2018), 1255–1264.
[29] Ren Zhang, Jianyu Zhang, Yu Chen, Nanhao Qin, Bingshuang Liu, and Yuan Zhang. 2011. Making eclipse attacks computationally infeasible in large-scale DHTs. In *Performance Computing and Communications Conference (IPCCC), 2011 IEEE 30th International*. IEEE, 1–8.
[30] Shijie Zhang and Jong-Hyouk Lee. 2019. Double-spending with a Sybil attack in the Bitcoin decentralized network. *IEEE Transactions on Industrial Informatics* 15, 10 (2019), 5715–5722.
[31] Quanyan Zhu and Tamer Basar. 2015. Game-theoretic methods for robustness, security, and resilience of cyberphysical control systems: games-in-games principle for optimal cross-layer resilient control systems. *IEEE control systems* 35, 1 (2015), 46–65.