IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 21, 2018, accepted November 3, 2018, date of publication December 4, 2018,
date of current version December 31, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2884764

A Game-Theoretic Analysis of Shard-Based Permissionless Blockchains

MOHAMMAD HOSSEIN MANSHAEI[1], MURTUZA JADLIWALA[2], ANINDYA MAITI[3], AND MAHDI FOOLADGAR[1]

[1]Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran
[2]Department of Computer Science, The University of Texas at San Antonio, San Antonio, TX 78249, USA
[3]Institute for Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249, USA

Corresponding author: Mohammad Hossein Manshaei (manshaei@cc.iut.ac.ir)

ABSTRACT Low transaction throughput and poor scalability are significant issues in public blockchain consensus protocols, such as Bitcoins. Recent research efforts in this direction have proposed shard-based consensus protocols where the key idea is to split the transactions among multiple committees (or shards), which then process these shards or set of transactions in parallel. Such a parallel processing of disjoint sets of transactions or shards by multiple committees significantly improves the overall scalability and transaction throughout of the system. However, one significant research gap is a lack of understanding of the strategic behavior of rational processors within committees in such shard-based consensus protocols. Such an understanding is critical for designing appropriate incentives that will *foster cooperation* within committees and *prevent free-riding*. In this paper, we address this research gap by analyzing the behavior of processors using a game-theoretic model, where each processor aims at maximizing its reward at a minimum cost of participating in the protocol. We first analyze the Nash equilibria in an N-player static game model of the sharding protocol. We show that depending on the reward sharing approach employed, processors can potentially increase their payoff by unilaterally behaving in a *defective* fashion, thus resulting in a social dilemma. In order to overcome this social dilemma, we propose a novel *incentive-compatible reward sharing* mechanism to promote cooperation among processors. Our numerical results show that achieving a majority of *cooperating* processors (required to ensure a healthy state of the blockchain network) is easier to achieve with the proposed incentive-compatible reward sharing mechanism than with other reward sharing mechanisms.

INDEX TERMS Sharding, blockchain, game theory, cooperation, and incentive design.

I. INTRODUCTION

A *blockchain* is an append-only, immutable distributed database that records a time-sequenced history of facts called *transactions*. Transactions are typically grouped into *blocks*, and the blockchain protocol enables the construction and maintenance of consistent copies of the cryptographic *hash-chain* of blocks in a distributed fashion. The first blockchain protocol was introduced in 2009 by Satoshi Nakamoto to support the *Bitcoin* cryptocurrency [1]. A key aspect of this protocol is the *consensus* algorithm (also sometimes referred in the literature as *Nakamoto consensus*) which enables agreement among a network of *processors* or *miners* on the state of the blockchain (identified by its cryptographic digest), under the assumption that a fraction of them could be malicious or faulty. In addition to this, as Bitcoin's blockchain is *permissionless*, i.e., no trusted infrastructure to establish verifiable identities for processors exists or is assumed, consensus on the blockchain's state cannot be achieved using standard distributed Byzantine fault-tolerant consensus algorithms in the literature. In such a permissionless setting, the blockchain protocol selects (randomly and in an unbiased fashion) one processor once every 10 minutes on average (*epoch*), and this selected processor gets the right to commit (or append) a new block onto the blockchain. The network (other processors) *implicitly* accept this block by building on top of it in the next epoch or reject it by building on top of some other block in the hash-chain.

The consensus in Bitcoin is thus *long-term*, i.e., a block is said to be included in the blockchain if it is part of the longest valid blockchain and has received a significant number of confirmations.[1] The Bitcoin protocol uses

[1]Number of blocks added on top of the block in question in the longest valid blockchain.

a *Proof-of-Work (PoW)* mechanism to select the leader (processor with the right to commit a block) in each epoch in an unbiased fashion, which is nothing but a *hash puzzle* that each processor attempts to solve - one that succeeds is selected and gets the right to propose the next block. As PoW involves significant computation, Bitcoin's protocol includes a *reward mechanism* to incentivize processors to compete (in a fair fashion) and to behave honestly. As of July 2018 [2], there were a total of 1624 cryptocurrencies, a significant number of which use the same code base as Bitcoin or are directly inspired by Bitcoin's distributed consensus algorithm. The use of blockchains and blockchain-based distributed consensus, however, is not just restricted to cryptocurrencies. Systems that can host and execute arbitrary distributed applications (commonly referred to as "*smart contracts*") over a single public permissionless hash-chain, for example, *Ethereum* [3], have also become popular. Such systems also employ a Bitcoin-like Proof-of-Work based consensus algorithm and a related cryptocurrency (e.g., Ether in Ethereum) to incentivize processors or miners to participate honestly in the consensus process.

Despite its tremendous popularity, one significant shortcoming of Bitcoin's consensus protocol (and of similar public permissionless blockchain systems) is its low transaction throughput and poor scalability. With an average inter-block time of 10 minutes and a maximum block size of 10 MB, Bitcoin's transaction rate is currently only 7 transactions per second [4]. Similarly, Ethereum can support only roughly 20 transactions per second. This is significantly lower than the transaction rates afforded by centralized transaction processing systems. For instance, PayPal can process more than 450 transactions per second while VisaNet can process anywhere between 1667 and 56,000 transactions per second [4]. It is clear that the current Bitcoin and Ethereum transaction rates are not sufficient for many practical applications, and thus, there have been significant efforts towards improving their transaction throughputs, for example, BIP [5] and Bitcoin-NG [6] for Bitcoin and Raiden [7] for Ethereum.

Similarly, there have been other significant efforts within the research community towards improving the transaction throughput and scalability of public permissionless blockchain protocols in general. One key outcome of this line of research is *sharding* [8]–[10], which proposes to periodically partition the network of processors (in an unbiased fashion) into smaller *committees*, each of which processes a disjoint set of transactions (also called a *shard*)[2] in parallel with other committees. As each committee is reasonably small, it can run a classical Byzantine consensus protocol such as PBFT [11] to agree on a set of transactions rather than the traditional Nakamoto consensus of Bitcoin, thus increasing the overall transaction throughput of the system. Although the idea of parallelizing the tasks of transaction processing and reaching consensus (on a set of transactions) by partitioning the processor network into committees is promising, existing sharding proposals [8]–[10] fail to clarify how processors will be incentivized to honestly participate and discharge their committee duties.

Two facts about existing sharding protocols are relevant to this discussion and should be highlighted: (i) the intra-committee consensus algorithms (e.g., PBFT) employed by existing protocols are inherently *fault-tolerant*, i.e., they will operate correctly even in the presence of a certain number of faulty or non-participating committee members, and (ii) the agreed (or consensus) set of transactions within each committee is required to be ratified (or signed) by only a majority of the committee members in order for those to be included into a block. Now as participation in committee tasks (such as transaction validation, signature creation, etc.) impose a *cost* on processors, it is possible that *rational* processors may choose not to participate in these tasks (and get away with it as the protocol may still succeed at the end) if their remuneration is not appropriately determined. For example, if each processor within a committee is equally remunerated, a rational processor may choose to *free-ride*, i.e., get paid without participating in any committee work. *In summary, one key research gap in this line of research is a lack of understanding of the strategic behavior of rational processors in shard-based consensus protocols for public permissionless blockchains.* Such an understanding is critical for designing appropriate incentives that will *foster cooperation* within committees and *prevent free-riding*. Our goal in this paper is to address this research gap.

In line with the above goal, we first model shard-based protocols, and the interaction between processors in such protocols, using a static non-cooperative game by systematically quantifying processor strategies in such a game and the resulting payoffs. We show that in such a setting, if the total reward (received at the end of the game when a new block is successfully committed to the blockchain) is equally or uniformly distributed among all the participating processors, then the resulting strategic interactions can be characterized using a game with social dilemma, such as a *public goods game*. Consequently, we show that not participating in the committee tasks (by all processors) is a Nash equilibrium of the game. We further show that it is impossible to enforce a cooperative Nash equilibrium in this setting unless certain improbable conditions are met. Hence, we extend the current game model by considering *fair sharing* of rewards, instead of equal sharing, where processors receive benefits only if they have cooperated within their shards. In this new system, we derive the Nash equilibria and conditions under which such an equilibrium can be achieved. Although this game is still a public goods game, we were able to establish conditions for achieving cooperation by processors towards executing the committee tasks in this game. These conditions can be derived and verified by processors before they decide on their strategy to *cooperate* or *defect* in the game. Our results show that it is possible to achieve a cooperative equilibium in such a fair reward sharing system. Finally, we design

[2]Note that the committees are working inside shards in these protocols. Hence, we use the two terms interchangeably in the paper.

the *incentive-compatible* reward sharing protocol that further improves upon the fair sharing protocol by introducing a shard coordinator who can guide individual processors to follow the optimal strategy (cooperate or defect), based on a preview of the shard's consensus status in each epoch. Our numerical analysis shows that the *incentive-compatible* protocol can outperform both the uniform and fair reward sharing protocols. To the best of our knowledge, this paper is the first to investigate the selfish behavior of processors, and its effect, in shard-based permissionless blockchains.

The rest of the paper is organized as follows. In Section II, we discuss the state of the art and present a generic system model for shard-based blockchain protocols, considering rational processors. In Section III, we present the game model and investigate all possible Nash equilibria under different reward sharing schemes. In Section IV, we describe the proposed *incentive-compatible* reward sharing protocol, followed by numerical evaluations presented in Section V. Related research efforts have been outlined in Section VI. We conclude the paper in Section VII.

II. SYSTEM MODEL

In this section, we first generically outline details of a shard-based approach for achieving consensus in permissionless blockchains. Then, we formally describe the various costs involved for processors participating in shard-based consensus protocols, followed by the notion of processor rationality (or selfishness) in such protocols.

A. SHARD-BASED CONSENSUS PROTOCOL

Consider a network of N processors participating in a public permissionless blockchain. Processors in such a network do not have an identity assigned by a trusted third-party or a public-key infrastructure, i.e., they use self-generated pseudonyms as transient identifiers. For simplifying the exposition, we assume that all processors are *similar* to each other in terms of *computational capabilities*. Further, we assume that all processors are *honest*, but *selfish* (more details on this will follow in section II-C).

Let time be divided into fixed-sized *epochs*. The network accepts *transactions* in *blocks*, i.e., at the end of each epoch the network accepts and commits a new block of transactions. Any block B is composed of (or can be partitioned into) k disjoint sets of transactions B_i, where B_i can be empty. Each such *disjoint set* B_i is referred to as a *shard* and can be defined based on some property(ies) of transactions within that set, for example, least significant bits of the transaction hash. The number of shards (k) is a variable quantity and can grow linearly with the size of the network. The network determines a binary *validation function* V, which takes as an input a transaction (belonging to any shard) and any other data representing the current state of the blockchain and outputs whether the input transaction is valid or not, and all processors have access to such a function V.

Given the network above, *sharding* is a *distributed consensus protocol* executed among a set of processors and which outputs at the end of each epoch a block B containing k disjoint shards B_i, such that, all honest processors (within the set of executing processors) agree on B with a very high probability and all transactions within B are valid (i.e., satisfy V). At a high level, the protocol does this by splitting the network of processors into multiple disjoint *committees*, where each committee processes (validates and agrees on) a separate shard (B_i). Now, lets summarize the main steps (Figure 1) involved in sharding as they occur in the classical Elastico [8] protocol. Below, we summarize the main steps involved in sharding by outlining a classical protocol called *Elastico* [8]. Recent research efforts such as Omniledger [10] provide some enhancements and additional functionalities to the original sharding proposal in Elastico, but the key idea of partitioning the transactions into disjoint shards and assigning a committee of processors to process each shard in parallel remains the same in all shard-based protocols.

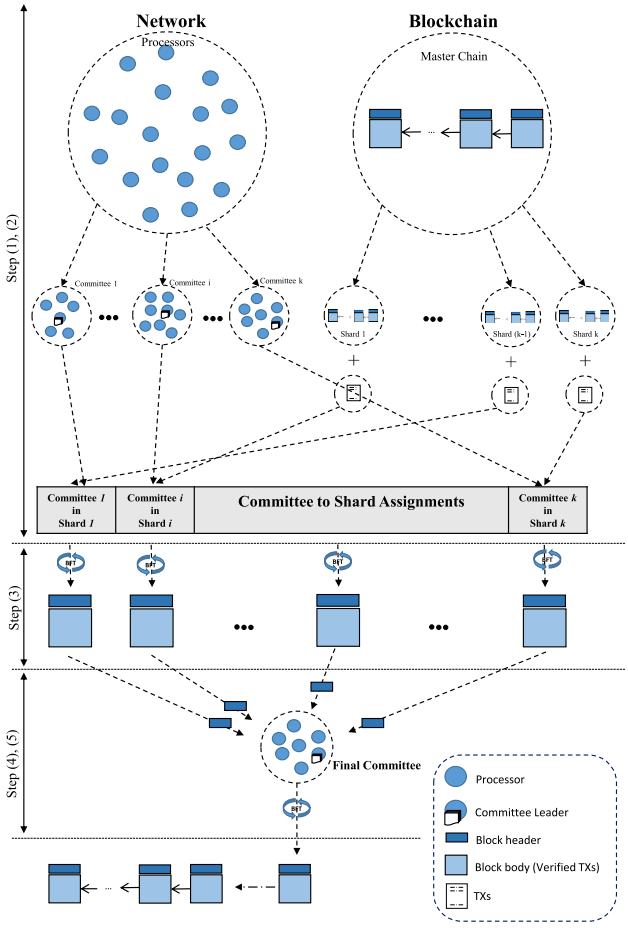


FIGURE 1. Conceptual view of a shard-based consensus protocol.

A sharding protocol proceeds in epochs and in each epoch the processors execute the following steps (in this order) [8]:

1) *Committee Formation:* First, each processor attempts to generate a publicly verifiable identity by solving

some *Proof-of-Work (PoW) puzzle*. In other words, each processor uses the solution of a PoW hash puzzle (i.e., the message digest that lies within the pre-determined target) as an identity in that epoch. There are two advantages of using a PoW puzzle for identity creation: (i) network (other processors) can verify the identity and, (ii) number of malicious sybils can be limited due to the computation involved in solving the puzzle. Each processor is then assigned to a committee corresponding to its established identity (say, using the s least significant bits of the identity). Moreover, each committee processes a distinct shard based on this s-bit identifier.

2) *Overlay Setup*: Next is the *community discovery* step where processors discover identities of other processors in their committee by communicating with each other. The outcome of this step is a *fully-connected overlay* for each committee in the network.
3) *Intra-Committee Consensus*: Next, processors run a standard *byzantine agreement protocol* such as *PBFT* [12] within their committees to agree on a set of transactions. Each committee then sends its consensus set of transactions B_i (or shard) to a *final committee* for inclusion in the new block B at the end of the current epoch. In order to be considered by the final committee, each shard B_i needs to be signed by a *simple majority*, i.e., by at least $\frac{c}{2} + 1$ processors for a committee of size c.
4) *Final Consensus*: A final committee (chosen based on a designated s-bit final committee identifier) then takes the consensus shards (B_i) from the previous step and merges these to create a final block B, creates a cryptographic digest or hash of B and broadcasts it to the rest of the network. During the merge operation, each processor in the final committee first validates that each shard B_i is signed by at least $\frac{c}{2}+1$ processors in the correct committee and then computes a union of all the shards to form the block B. After each processor in the final committee computes a union in this fashion, they then collectively run a byzantine agreement protocol such as PBFT [12] to arrive at a consensus on the final block B. The cryptographic digest of the final consensus block B needs to be signed by a simple majority of the final committee before it can be broadcast on the network.
5) *Randomness Generation for Next Epoch*: In the final step of the protocol, the final committee generates a set of *random strings* and broadcasts it to the network. These random strings are used by the processors in the identity creation and committee formation tasks of the *next epoch*.

B. PROCESSOR COSTS

We now characterize the costs (including, computation and communication costs) borne by the processors in each time epoch due to their participation in the sharding protocol. It should be noted that our goal here is not to arrive at a precise quantification of these costs, rather to characterize them such that they could be used to analyze the strategic behavior of processors while participating in the protocol. The protocol steps in each epoch, as outlined in the previous section, can be basically grouped into two phases: (1) *organization phase* and, (2) *committee participation phase*. During the organization phase, processors create identities using PoW puzzles, form committees and identify other processors in their committee (i.e., execute steps 1 and 2 in the protocol above). In the committee participation phase the processors validate their respective shards and arrive at an agreement with other committee members (i.e., execute steps 3, 4 and 5 in the protocol above).

It should be clear that the organization phase facilitates the committee participation phase, and is mandatory, i.e., if a processor does not does not have an identity and gets assigned to a committee, it cannot participate in committee-related tasks. Similarly, it should also be clear that the committee participation phase is not mandatory for processors, i.e., a processor could choose to create a verifiable identity and be assigned to a committee, but may choose not to participate in tasks such as shard validation and intra-committee consensus. If a subset of processors do not take part in the committee participation phase, it does not mean that the protocol will fail. The inherent fault-tolerance of intra-committee consensus protocols such as PBFT, and the simple majority rule employed in intra-committee voting, implies that a certain number of non-participation can be tolerated by the protocol. For the sake of generalization, we assume that if less than τ processors within a committee of size c do not participate in the committee participation phase, the entire protocol for that epoch fails, i.e., no new block is proposed in that epoch.

Thus, we can characterize the total cost for a processor to participate in an epoch of the sharding protocol based on the cost for executing the above two phases. For the organization phase, let's assume that a processor bears a cost c^m, which we refer to as the *mandatory cost*. It should be noted that c^m is a fixed cost and is independent of the number of transactions processed by the processor. Moreover, as solving the PoW puzzle is the most significant activity during the organization phase, c^m can be approximated using the current difficulty of the PoW puzzle and the average computational power of all the processors.

Accordingly, for executing the committee participation phase let's assume that a processor bears an *optional cost* c^o, depending on whether the processor fully participates in it or not. Unlike the mandatory cost, the optional cost c^o has two components: (i) a *fixed component* and, (ii) a *transaction-dependent component*. During the committee participation phase, a processor performs activities such as participation in intra-committee consensus the cost of which can be bounded by a fixed average cost [12]. We represent all these per-processor fixed costs during the committee participation phase as c^f. Another activity during this phase that all processors are expected to perform is verifying the validity of

all outstanding transactions (they have received) within their respective shards by using the validation function V. Depending on the complexity of the validation function V, this can be a significant cost (to a processor) which also depends on the number of outstanding transactions being validated. We represent the cost to validate each transaction using V by c^v. Hence, we can compute the total optional cost c_i^o for a processor P_i as:

$$c_i^o = c^f + |x_i^j| c^v \tag{1}$$

where x_i^j is the vector of transactions received and validated by processor P_i. The average per-processor cost (c_i^t) for participation in each epoch of the shard-based protocol can thus be characterized as $c_i^t = c^m + c^f + |x_i^j| c^v$.

One point that needs further clarification is why a processor may choose not to execute the committee participation phase after executing the organization phase. Our rationality assumption, which we describe next, provides this clarification.

C. RATIONALITY ASSUMPTION

Earlier research efforts on sharding [8], [10] have assumed a *byzantine adversary* where processors controlled by the adversary can be *arbitrarily malicious*, i.e., malicious processors could arbitrarily deviate from the correct execution of the protocol or could arbitrarily drop protocol messages. In this work, however, we assume that processors are *honest* but *selfish*. In other words, processors do not arbitrarily deviate from protocol execution or drop protocol messages, but decide against participation in the protocol only when there is an incentive (financial or otherwise) to do so. Let us further provide a brief intuition of the notion of rationality in this setup. All processors receive some rewards if the protocol execution in an epoch is successful, for example, in terms of block rewards, transaction fees, etc. The precise nature of rewards depend on the specific system or application that the blockchain protocol enables. Moreover, as discussed in the earlier section, all processors bear some costs for fully participating in both phases of the protocol. The total *benefit* or *payoff* received by processors in each epoch is the difference between the obtained reward and the spent costs in that epoch. A selfish (or rational) processor will always choose a protocol participation strategy that improves its benefit or payoff. If a processor does not execute the organization phase, it does not get any reward as it is not a part of any committee. However, a rational processor's strategy could be to execute the organization phase but refrain from the committee participation phase. Such a selfish strategy saves on the optional cost c^o and may result in a reward if enough other processors participate fully, and thus may provide more benefit or payoff to the rational processor. We assume that a rational processor will always choose such a selfish strategy which provides more benefit or payoff, if it exists. In summary, the goal of each processor is to maximize its individual payoff (received at the end of each epoch), without maliciously trying to deviate or disrupt the protocol. We also assume that processors do not collude/coordinate in order to jointly maximize their combined utility.

TABLE 1. List of symbols.

Symbol	Definition
k	Number of shards (or committees)
N	Number of processors
x_i^j	Vector of received transactions by processor i in shard j
y^j	Vector of transactions submitted by shard j to Blockchain
c	Minimum number of processors in each committee
τ	Required number of processors in shard for consensus
r	The benefit for each transaction
b_i	Benefit of processor i after adding the block
c_i^t	Total cost of computation for processor i
c^o	Total optional costs in each epoch
c^m	Mandatory costs in each epoch to enter the shard
c^v	Cost of transaction verification
c^f	Fixed costs in optional cost
BR	Block Reward
l_j	Number of cooperative processors in each shard
L	Total number of cooperative processors in all shards
$C_j^{l_j}$	The set of all cooperative processors in shard j
$D_j^{n-l_j}$	The set of all defective processors in shard j
C^L	The set of all cooperative processors
D^{N-L}	The set of all defective processors

III. SHARD-BASED BLOCKCHAIN GAME

In this section, we present the game-theoretic aspects of a shard-based blockchain protocol with multiple processors in an honest but selfish environment. We first introduce a non-cooperative N-Player game model that we refer to as the *shard-based blockchain game* $\mathbb{G}$. Upon starting an epoch t, processors must decide whether to collaborate with each other, verify transactions, and make a block to be appended to the chain (i.e., take part in the community participation phase), after the organization phase as we addressed in the previous section. The key point of the game-theoretic analysis is to consider the computation costs for processors who verify transactions and participate in consensus mechanism, as presented in Section II-B and II-C, and the total benefits when they agree on a valid block. Therefore, using a game-theoretic analysis, we investigate whether block generation can emerge in such a non-cooperative system. By means of our game model and the related analysis, we would like to show that with a uniform distribution of rewards in these protocols, the interactions between processors fall in a category of games, where there exists a social dilemma of all-defection behavior. We then propose a novel reward sharing protocol and address the conditions for having a new class of equilibrium, where a subset of processors will be forced to cooperate. Table 1 summarizes the notation used throughout the paper.

A. GAME MODEL

Game theory allows for modeling situations of conflict and for predicting the behavior of participants when they interact with each other. In our *shard-based blockchain game* $\mathbb{G}$, processors must decide upon joining the shards whether to cooperate and contribute to optional costs (as addressed in Section II-B) or not. We model $\mathbb{G}$ as a static game, because all processors must choose their strategy simultaneously, after they have join their shard. This modeling decision also keeps our analysis tractable, while conforming to a simple model of processor rationality. The game $\mathbb{G}$ is defined as a triplet $(\mathcal{P}, \mathcal{S}, \mathcal{U})$, where $\mathcal{P}$ is the set of players, $\mathcal{S}$ is the set of strategies and $\mathcal{U}$ is the set of payoff values. We also assume that at any time epoch t, a game is played among all the processors in all shards, because the benefits of successfully adding a block is shared among all processors.

• **Players** $(\mathcal{P})$**:** The set of players $\mathcal{P} = \{P_i\}_{i=1}^{N}$ corresponds to the set of processors who have already joined shards in a given epoch time t. In fact, all N processors must have already performed PoW and paid the mandatory costs c^m. Considering the number of shards in our system model, i.e., k, we conclude that each shard has $n = N/k$ committee members. During this epoch time in our game, we assume that each processor P_i in shard j receives the vector x_i^j of transactions to verify and participate in the consensus algorithm.

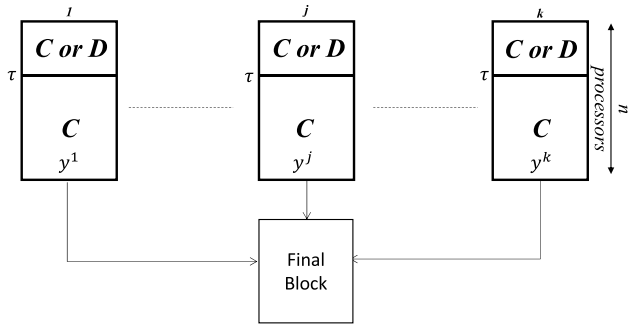


FIGURE 2. In each shard at least τ processors among n processors must be cooperative to perform consensus algorithm. Each Shard j submits the final y^j vector of transactions to make the final block.

As it is shown in Figure 2, we also assume that to perform a consensus algorithm in each shard we need at least τ processors who agree on a given list of transactions. For example, in Elastico protocol which uses PBFT, τ is equal to $\frac{2}{3}n$. Finally, y^j, $j \in \{1, \ldots, k\}$ represents the result of the consensus algorithm including the list of transactions that would be added to the blockchain by shard j.

• **Strategy** $(\mathcal{S})$**:** Each processor P_i can choose between two moves s_i: (i) *Cooperate* C, or *(ii) Defect* D. Hence the set of strategies in this game is $\mathcal{S} = \{C, D\}$. The strategy of processor P_i determines whether P_i participates in all optional tasks presented in Section II or not. In particular, if processor P_i plays C, it will accept and verify all received transactions. In this case, it also cooperates in all consensus algorithms and incurs cost c^o for its participation. Contrary to a cooperative behavior, a given processor can refuse all transaction verifications and simply do nothing during the community participation phase (i.e., play D).

• **Payoff** $(\mathcal{U})$**:** After executing the protocol and inserting a new block to the hash-chain at the end of each epoch, we assume that the network of participating processors receive two types of *rewards*. This assumption is motivated from observation in current permissionless blockchain applications such as Bitcoin [1] and Ethereum [3]. The first is a *fixed* reward for adding a new block, called the *block reward* (BR). The current block reward for Bitcoin, for example, is 12.5 BTC [13]. The second *variable* reward is the sum of *transaction fees* of all transactions within the accepted block. For simplicity, let's assume that each transaction includes an average *fee* r. Hence, the reward or benefit that a given shard j can receive from transaction fees is $r|y^j|$, while the total transaction fee reward due to the appended block in each epoch can be estimated as $TF = r\sum_{j=1}^{k}|y^j|$.

Recall that the total cost of cooperation for processor P_i is equal to $c_i^t = c^m + c_i^o = c^m + c^f + |x_i^j|c^v$ if the processor acts honestly and follows the protocol. All processors should pay the mandatory costs, i.e., c^m in order to be in a committee and finally receive the reward. But they can avoid paying optional cost c^o, including the cost of verifications. In summary, we can divide processors into two groups of cooperative and defective processors, based on whether they contribute to optional tasks (i.e., play C and pay c^t) or not (i.e., play D and pay only c^m). Let $\mathcal{C}_j^{l_j}$ and $\mathcal{D}_j^{n-l_j}$ denote the sets of l_j cooperating players and $n - l_j$ defecting players in a given shard j with n processors. Recall that in order to obtain a consensus transaction vector y^j in a given shard j, $|\mathcal{C}_j^{l_j}|$ must be greater than or equal to τ $(|\mathcal{C}_j^{l_j}| \geq \tau)$. It should be noted that we implicitly assume that each shard provides a non-empty y^j. Due to lack of clarity in current sharding proposals [8]–[10], if one or more shards fail to provide a y^j in an epoch we assume that the network cannot compute and append a new block in that epoch.

If we assume that all processors receive an equal share of profits after block computation (i.e., the existing protocol), we can calculate the reward share for each processor as:

$$\frac{BR + r\sum_{i=1}^{k}|y^j|}{N} \tag{2}$$

In other words, all processors receive an equal share of the rewards from block reward and total transaction fees. Hence, if we assume that a processor P_i was cooperative, i.e. $P_i \in \mathcal{C}_j^{l_j}$, we can compute the payoff of processor P_i in shard j as:

$$u_i^j(C) = b_i - c_i^t = \frac{BR + r\sum_{j=1}^{k}|y^j|}{N} - (c^m + c^f + |x_i^j|c^v) \tag{3}$$

Similarly, if P_i is defective, i.e., $P_i \in \mathcal{D}_j^{n-l_j}$, its payoff would be:

$$u_i^j(D) = \frac{BR + r\sum_{j=1}^{k}|y^j|}{N} - c^m \tag{4}$$

Considering the above calculated payoffs, we analyze the game $\mathbb{G}$ next.

B. GAME ANALYSIS

In order to get an insight into the strategic behavior of the processors, we apply the most fundamental game-theoretic concept, named Nash equilibrium, introduced by John Nash [14]:

Definition 1: In a Nash equilibrium strategy profile, none of the players can unilaterally change its strategy to increase its utility.

In other words, if in a non-cooperative game all strategies are mutual best responses to each other, then no player has any motivation to deviate unilaterally from the given strategy profile. Nash also proved that any finite game has at least one Nash equilibrium strategy profile. In non-cooperative game theory, *Prisoner's dilemma* or PD, discovered by Flood and Dresher in 1950 and later formalized by Tucker [15], is a classical 2-player game which shows why two rational individuals might not cooperate, even if it appears that the cooperative strategy is more beneficial for both of them (i.e., Pareto Optimality). In PD, each individual has two strategies of *cooperation* and *defection*, and the defection strategy strictly dominates the cooperation strategy. Hence, the only Nash equilibrium in PD, is a mutual defection.

More than 20 years later, Hamburger defined the analogous N-player version of PD game in [16]. This extension is called *public good game* (PGG). In a PGG setting, each individual can cooperate and pay a contribution of α or defect and do not pay anything. Then all contributions would be summed and multiplied by a reward factor $\gamma > 1$. Finally, the total reward would be distributed among all users equally, whether they have cooperated or defected. In other words, if n agents out of N cooperate, their payoff would be $\frac{\gamma\alpha n}{N} - \alpha$ and the defectors' payoff is $\frac{\gamma\alpha n}{N}$. Indeed, the total payoff of all users is maximized when everyone contributes to the public good. However, it has been proved that the Nash equilibrium in this game is defection by all users. A complete survey of PGGs and related results is available in [17].

Following our definition for *shard-based blockchain game* $\mathbb{G}$, we show in the following theorem that $\mathbb{G}$ is a PGG. In other words, the system fails to make any new block and remain in the same state if all processors defect initially.

Theorem 1: In each epoch of a shard-based blockchain game $\mathbb{G}$ with N processors, if rewards are equally shared among all processors, then $\mathbb{G}$ reduces to a public goods game.

Proof: Let us consider the strategy profile where all processors defect and do not pay optional cost c^o after joining to the shards. We call this strategy profile $All-D$. The payoff of each processor i would be then $u_i = -c^m$. In this case, none of the processors can unilaterally change its strategy to increase its payoff. Because, the only cooperative processor cannot obtain any reward without the contribution of at least $\tau - 1$ other processors in its shard, as addresses in Section II. In other words, the new payoff of each processor who deviates would be $-c^m - c^f - |x_i^j|c^v$ which is indeed smaller than $-c^m$. Hence, $All-D$ is a Nash equilibrium profile in this game and $\mathbb{G}$ is a PGG. □

Theorem 2 further shows that we can never enforce an all-cooperation strategy ($All - C$) in the game $\mathbb{G}$, as it is not a Nash Equilibrium.

Theorem 2: In each epoch of a shard-based blockchain game $\mathbb{G}$ with N processors, if rewards are equally shared among all processors, we cannot establish All-Cooperation strategy profile as a Nash equilibrium.

Proof: We first assume that all N processors have already cooperated in transaction verifications (i.e., $All - C$ strategy profile) and paid the optional cost c^o. We can compute the payoff of each processor P_i by Equation (3). Hence, if a given processor deviates from the cooperation and plays defection unilaterally, its payoff would be equal to Equation (4), which is always greater than cooperative payoffs at Equation (3). Hence, each user has incentive to deviate unilaterally and increases its payoff. Then, the $All - C$ strategy profile is never a Nash equilibrium. □

Finally, Theorem 3 shows the conditions under which we can enforce an equilibrium in game $\mathbb{G}$, where some processors cooperate.

Theorem 3: Let $\mathcal{C}_j^{l_j}$ and $\mathcal{D}_j^{n-l_j}$ denote the sets of l_j cooperating processors and $n - l_j$ defecting processors inside each shard j with n processors. If $L = \sum_{j=1}^{k} l_j$ is the total number of cooperative processors, $(\mathcal{C}^L, \mathcal{D}^{N-L})$ represents a Nash equilibrium profile in each epoch of the game $\mathbb{G}$, if and only if $l_j = \tau$ in all shards j, where $\mathcal{C}^L = \bigcup_j \mathcal{C}_j^{l_j}$ and $\mathcal{D}^{N-L} = \bigcup_j \mathcal{D}_j^{n-l_j}$.

Proof: If in all shards, there exist exactly $l_j = \tau$ cooperative processors, any cooperative processor cannot deviate unilaterally to increase its payoff. This is because the deviation will result in failed consensus, and remove block reward BR and $r|y^j|$ transaction fee from the benefits. Consequently, the cooperative processor's payoff would be decreased. Moreover, similar to previous cases, there is no incentive to deviate for defective processors, since they must incur additional costs (on top of c^m) for their cooperation, while this will not change the result of the consensus algorithm. □

The above theorems prove that if rewards are uniformly distributed among processors, a cooperative equilibrium cannot be enforced in shard-based public permissionless blockchains. Hence, in the following section we define a new reward sharing approach, which promotes cooperation among processors by providing appropriate incentives.

C. FAIR REWARD SHARING

In this section, we extend our game model to include a fair reward sharing approach, where each processor receives a reward if and only if it has already cooperated with other processors within the shard. Let's call this new game $\mathbb{G}^F$, in which the payoff of cooperative processors in set $\mathcal{C}_j^{l_j}$ is:

$$u_i^j(C) = \frac{BR}{kl_j} + \frac{r|y^j|}{l_j} - (c^m + c^f + |x_i^j|c^v), \qquad (5)$$

Recall that we assume $l_j \geq \tau$ for the consensus algorithm and each shard j will submit a non-empty y^j set to the blockchain. Analysis of the case where the processors cannot make a consensus on a given vector of transactions can be easily extended from our model, by assigning BR and r a value of zero (no benefits). As Equation (5) shows, we first assume that the BR is uniformly distributed among shards and each cooperative processor can receive a share of it. Moreover, each shard j receives all fees for all transactions that it has submitted to the blockchain. Then, in each shard this reward is uniformly distributed among all cooperative processors. It is worth mentioning that $|x_i^j|$ may not always be equal to $|y^j|$. It means that a processor P_i might be cooperative but finally all other processors may agree on a vector of transactions y^j that is different from x_i^j. Thus, contrary to the standard shard-based protocols, in $\mathbb{G}^F$ the defective processors' payoff can be calculated as:

$$u_i^j(D) = -c^m, \tag{6}$$

because the defective processors will not receive any benefit. It is easy to show that the conditions of Theorem 1 still hold in the new game $\mathbb{G}^F$ and the game $\mathbb{G}^F$ is PGG. However, we can show that in this newly defined game $\mathbb{G}$, it is easier to enforce users to cooperate at a Nash equilibrium profile. We derive the conditions under which there exists a cooperative Nash equilibrium profile in game $\mathbb{G}^F$, with the following theorem.

Theorem 4: Let $\mathcal{C}_j^{l_j}$ and $\mathcal{D}_j^{n-l_j}$ denote the sets of l_j cooperating processors and $n - l_j$ defecting processors inside each shard j with n processors, respectively. $(\mathcal{C}^L, \mathcal{D}^{N-L})$ represents a Nash equilibrium profile in each epoch of game $\mathbb{G}^F$, if the following conditions are satisfied:

1) *In all shards j, $l_j \geq \tau$.*
2) *If for a given processor P_i in shard j, $x_i^j = y^j$, then the number of transactions $|x_i^j|$ must be greater than $\theta_c^1 = \frac{c^f - \frac{BR}{kl_j}}{r/l_j - c^v}$*
3) *If for a given processor P_i in shard j, $x_i^j \neq y^j$, then the number of transactions $|x_i^j|$ must be smaller than $\theta_c^2 = \frac{\frac{BR}{kl_j} + \frac{r|y^j|}{l_j} - c^f}{c^v}$.*

Proof: The number of cooperative processors must be greater than the consensus threshold τ, otherwise the cooperative processors will not receive any transaction and block reward benefits for their cooperation. Hence, they can increase their payoff by unilaterally deviating from the cooperative strategy. We now find the largest group of cooperative processors l_j in each shard, where no processor in $\mathcal{D}_j^{n-l_j}$ can join $\mathcal{C}_j^{l_j}$ to increase its payoff. Let's assume that l_j^* is this largest set of processors. If processor P_i^j is among the set of cooperative processors, then it will not unilaterally deviate if its payoff (calculated by Equation (5)) is greater than $-c^m$. Two possible cases could happen in this case. First, P_i^j could be among processors who have the same vector of transactions as the output of the shard, i.e., $x_i^j = y^j$. In this case, P_i^j will not deviate from cooperation if:

$$\frac{BR}{kl_j} + \frac{r|x_i^j|}{l_j} - (c^m + c^f + |x_i^j|c^v) \geq -c^m, \tag{7}$$

which shows that $x_i^j \geq \theta_c^1$, where

$$\theta_c^1 = \frac{c^f - \frac{BR}{kl_j}}{r/l_j - c^v}.$$

In the second case, processor P_i^j have cooperated with others in $\mathcal{C}_j^{l_j}$, but its vector of transactions is different from the output of the shard, i.e., $x_i^j \neq y^j$. Hence, the following condition must be satisfied if this user wants to remain in the cooperative set.

$$\frac{BR}{kl_j} + \frac{r|y^j|}{l_j} - (c^m + c^f + |x_i^j|c^v) \geq -c^m, \tag{8}$$

which shows that $x_i^j < \theta_c^2$, where

$$\theta_c^2 = \frac{\frac{BR}{kl_j} + \frac{r|y^j|}{l_j} - c^f}{c^v}.$$

If l_j^* represents the largest set of cooperative processors in each shard, then $(\mathcal{C}^L, \mathcal{D}^{N-L})$ would be the unique cooperative Nash equilibrium of the game $\mathbb{G}^F$. Please note that this NE is a unique cooperative equilibrium of the game, as we have already found the largest set of cooperative processors in all shards. □

Note that by increasing the optional costs of computation (whether c^f is in the numerator or c^v in denominator of θ_C) and for any given number of transactions $|x_i^j|$, processors will be tempted to be more defective as the threshold θ_c^1 will be increased. This is in line with our intuition that processors are not cooperative if the cost of cooperation is high. On the other hand, the calculated threshold shows that by increasing the number of processors N and consequently the number of shards k, the processors would be more defective. This is representing the case where the processors will not cooperate in the hope that other processors will participate in the transaction verifications and other optional tasks in the defined protocol. Moreover, as block reward is now shared among a larger number of processors, processor rewards from cooperation are reduced resulting in a reduced overall payoff, and thus less incentive for a processor to cooperate. *In summary, our results so far demonstrate that $\mathbb{G}^F$ is a PGG with defection by processors as one potential equilibria, resulting in the system/network not adding any new blocks. However, under certain conditions where the number of transactions are large enough for processors, we have demonstrated that $\mathbb{G}^F$'s fair reward distribution approach can provide enough incentives to enable the level of processor cooperation required to successfully add a new block at the end of each epoch.* Next, we apply the results from Theorem 4 to design an *incentive-compatible* sharding protocol for public permissionless blockchain.

IV. INCENTIVE-COMPATIBLE REWARD SHARING

Our next goal is to extend the current shard-based consensus protocols by considering the strategic behavior (optimal strategies) of rational or selfish processors during protocol participation such that the required level of cooperation can be enforced in these protocols. The fair reward sharing based game model, and the related analysis, provides us with some insights on how to design such an *incentive-compatible shard-based consensus protocol*. However, there are two significant challenges that need to be overcome before applying these game-theoretic results (specifically, Theorem 4) towards the design of such an incentive-compatible protocol. The first challenge is to determine how to enforce, and who will enforce, cooperation in the distributed computing environment of the protocol? Second, as the optimal strategy of each processor (to cooperate or defect) depends on the number of received transactions compared to a fixed threshold and whether that transaction set is part of the final consensus (Theorem 4), how can one determine the optimal strategy for a processor prior to the consensus taking place? We address both of these challenges in our proposed *incentive-compatible* reward sharing protocol, discussed next.

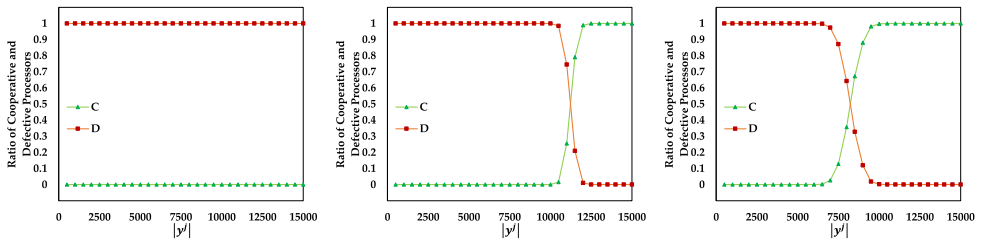
With regards to the first challenge, one way cooperation can be enforced is by means of a "*coordinator*" (in each shard) who will announce to the processors whether cooperation would be in their interest or not, i.e., whether they should cooperate in the upcoming epoch or not. Coordinators for each shard could be either randomly selected from among the processors (in each shard) or it could also be a centralized trusted entity. Besides announcing cooperate/defect decisions for each processor in the shard (based on the information received from each processor), the coordinator also enforces compliance to these strategy decisions by appropriately rewarding processors who comply and punishing processors who do not. This reward sharing or punishment is carried out according to the fair reward sharing strategy discussed earlier. Regarding the second challenge, where the coordinator needs to efficiently obtain transaction information from each processor to determine their optimal strategy in the shard, one straightforward solution is for each processor to share a *hash* or *message digest* of their current transaction set x_i^j by employing a cryptographically secure hash function.

Based on the above insights, our proposed incentive-compatible protocol (Algorithm 1) would proceed as follows: In each epoch, processors will first attempt to obtain an *ID* to participate in a shard or committee by solving the PoW puzzle. After committee formation and assignment phases, each processor P_i in shard j obtains a set of transactions x_i^j to verify. At this stage, all processors calculate $H(x_i^j)$ and submit it to a pre-elected coordinator, where H is a cryptographically secure hash function. The coordinator then finds the maximum subset of processors with similar $H(x_i^j)$ and uses it to estimate l_j and the thresholds θ_c^1 and θ_c^2 (according to Theorem 4). Based on this information, the coordinator also determines the set of potential cooperative and defective processors in that epoch. The coordinator then assists the processors in selecting an optimal strategy for themselves by *publicly announcing* the *expected* set of cooperative processors and the computed thresholds θ_c^1 and θ_c^2.

Each processor P_i in shard j which is present in the expected cooperative processor set publicly announced by the coordinator then makes a cooperation or defection decision by comparing the size of its transaction set x_i^j to the announced threshold θ_c^1. Similarly, each processor P_i in shard j which is *not* in the (expected) cooperative processor set announced by the coordinator makes a cooperation or

Algorithm 1 Incentive-Compatible Protocol

procedure Initialization and Committee Creation
  $ID, Shard \leftarrow ComputeID(epochRandomness, IP, PK)$
  $x_i \leftarrow ShardTransactions(Shard)$
end procedure
procedure Cooperaive/Defective Processor Selection
  P_i sends $H(x_i^j)$ to *Coordinator*
  if *Coordinator* **then**
    Receive $H(x_i^j)$s
    $l_j \leftarrow$ Maximum number of processors with common transactions
    if $l_j < \tau$ **then**
      return $All - D$
    else
      Prepare the list of l_j processors $\mathcal{C}_j^{l_j}$
      Calculate θ_c^1 and θ_c^2 from *Theorem 4*
      return θ_c^1, θ_c^2, and $\mathcal{C}_j^{l_j}$
    end if
  end if
end procedure
procedure Shard Participation (Consensus)
  if $P_i \in \mathcal{C}_j^{l_j}$ and $|x_i^j| \leq \theta_c^1$ **then**
    return Defect
  else if $P_i \notin \mathcal{C}_j^{l_j}$ and $|x_i^j| \geq \theta_c^2$ **then**
    return Defect
  end if
  Verify transactions and create a set of verified transactions y^j by all remaining cooperative processors
  Consensus on verified transactions
  Sign BFT agreement result
  return Signature, Agreed block's header
end procedure
procedure Verification, Reward, and Punishment
  Verify whether $P_i \in C^L$ have cooperated in each shard
  Distribute rewards among cooperative P_i according to Equation (5)
end procedure



BR	1000
c^m	10
c^f	6
c^v	0.0005
r	0.1
$P(x_i^j \neq y^j)$	15%
N	$\approx$ 3000
n	$\approx$ 100
$\lvert y^j \rvert$	$\approx$ 500-15000

(d)

FIGURE 3. Ratio of *cooperative* and *defective* processors for different sizes of y^j. (a) Uniform. (b) Fair. (c) Incentive-Compatible. (d) Simulation Parameters.

defection decision by comparing the size of its transaction set x_i^j to the announced threshold θ_c^2. At the end of committee participation phase, if the protocol is successful the coordinator distributes the end-of-epoch block and transaction rewards as defined by the fair reward distribution scheme (Equations 5 and 6). It is easy for the coordinator to verify at the end of the epoch if a given processor P_i behaved in an incentive-compatible fashion by following its recommendation - any processor that deviates from the coordinator's recommended cooperation strategy will not receive any share of the rewards obtained at the end of the epoch (if the protocol is successful in adding a block). This acts as an implicit *punishment* to deter processors from deviating from the recommended strategy.

V. NUMERICAL ANALYSIS

We conduct a comprehensive set of numerical simulations, in order to validate how our proposed *incentive-compatible* protocol compares with uniform and fair reward sharing protocols in shard-based blockchains. We first detail the experimental setup used to simulate a basic shard-based blockchain in Section V-A. Variants of the simulation were used to analyze multiple parameters that may affect the strategy of individual processors, and thereby its effect on the successful operation of the blockchain network.

A. EXPERIMENTAL SETUP

We simulate a shard-based public permissionless blockchain with approximately N ($\pm 1\%$) processors that are selfishly following a protocol to reach consensus in each shard, then combine all shards to add the next block, and finally collect their reward at the end of each epoch. We assume committees of size 100 ($\pm 1\%$), and the required number of processors in each shard for consensus is $\tau \approx 51$. Also, the number of committees (and shards) grow linearly *w.r.t.* the number of processors in the network ($k \approx \frac{N}{100}$). Each processor in the network is assumed to receive $|x_i^j| \approx |y^j|$ ($\pm 1\%$) transactions corresponding to the shard it belongs to. As imperfect views of the network is common occurrence in real-world networks, we also assume that the number of processors with $x_i^j \neq y^j$ is approximately 15%. We present mean results of 100 iterations for each combination of parameters (in Figures 3-6), i.e., every point in the graphs was obtained after averaging the results of 100 independent epochs with that particular set of parameters.

B. NUMBER OF TRANSACTIONS

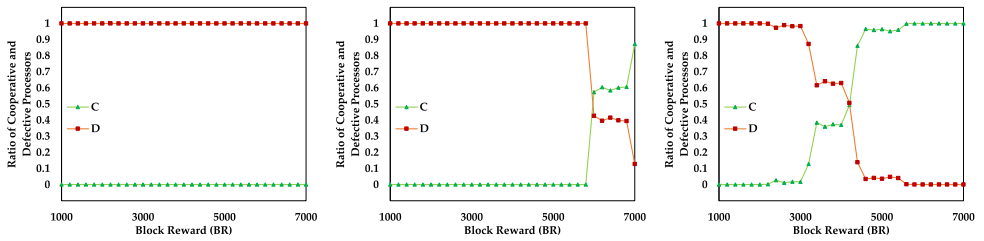
We first analyze the effect of varying the average number of transactions $|x_i^j|$ between 500 and 15000. The corresponding ratios of cooperative and defective processors are plotted in Figure 3. As intuitive, the uniform reward sharing results in all defect (Figure 3a), and thus no block is ever added to the blockchain. In case of fair and incentive-compatible reward sharing protocols (Figure 3b and 3c, respectively) we observe that processors opt for all defect strategy when the number of transactions is low, but eventually change their strategy to cooperate as the number of transactions becomes high enough to make a profit. More importantly, the proposed incentive-compatible reward sharing protocol achieves a majority of cooperative processors for lesser number of transaction than in the case of fair sharing, which is favorable.

C. BLOCK REWARD

We next analyze the effect of varying the block reward BR between 1000 and 7000, and the corresponding ratios of cooperative and defective processors are plotted in Figure 4. As before, the uniform reward sharing results in all defect (Figure 4a), regardless of the value of the block reward. In case of fair and incentive-compatible reward sharing protocols (Figure 4b and 4c, respectively) we observe that processors opt for all defect strategy when the block reward is low, but eventually change their strategy to cooperate as the block reward gets high enough to make a profit. Again, the proposed incentive-compatible reward sharing protocol achieves a majority of cooperative processors for lesser valued block reward than in the case of fair sharing, which is favorable.

D. SIZE OF THE NETWORK

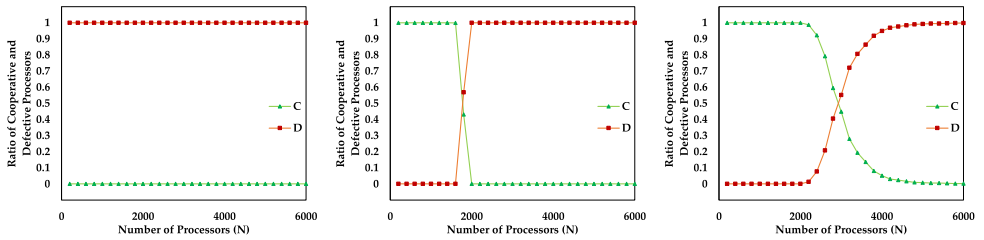
The number of processors in the network in a given epoch can vastly impact the strategy for individual processors, because if a small reward is shared between a large number of cooperative processors, it may not cover other costs associated with



BR	1000-7000
c^m	10
c^f	6
c^v	0.001
r	0.1
$P(x_i^j \neq y^j)$	15%
N	$\approx$ 1000
n	$\approx$ 100
$\lvert y^j \rvert$	$\approx$ 10000

(d)

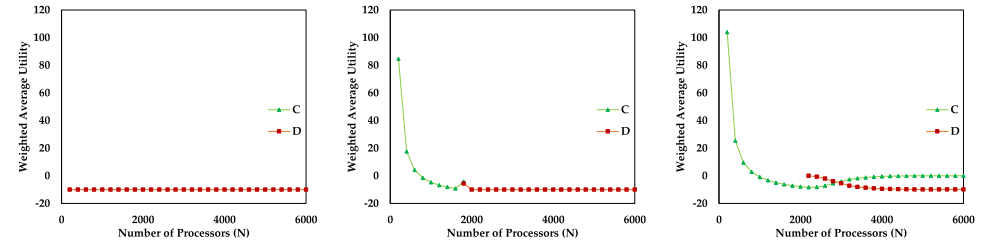
FIGURE 4. Ratio of *cooperative* and *defective* processors for different values of *BR*. (a) Uniform. (b) Fair. (c) Incentive-Compatible. (d) Simulation Parameters.



BR	10000
c^m	10
c^f	6
c^v	0.001
r	0.1
$P(x_i^j \neq y^j)$	15%
N	$\approx$ 100-6000
n	$\approx$ 100
$\lvert y^j \rvert$	$\approx$ 10000

(d)

FIGURE 5. Ratio of *cooperative* and *defective* processors for different values of *N*. (a) Uniform. (b) Fair. (c) Incentive-Compatible. (d) Simulation Parameters.



BR	10000
c^m	10
c^f	6
c^v	0.001
r	0.1
$P(x_i^j \neq y^j)$	15%
N	$\approx$ 100-6000
n	$\approx$ 100
$\lvert y^j \rvert$	$\approx$ 10000

(d)

FIGURE 6. Weighted average utility of *cooperative* and *defective* processors for different values of *N*. (a) Uniform. (b) Fair. (c) Incentive-Compatible. (d) Simulation Parameters.

participation (such as c^f). We observe this intuition in effect in Figure 5, where N is varied between 100 and 6000. Both the proposed incentive-compatible and fair reward sharing protocols lose majority of cooperative processors when N is increased significantly. However, the proposed incentive-compatible reward sharing protocol retains a majority of cooperative processors for greater number of processors than in the case of fair sharing, which is desirable. As before, the uniform reward sharing results in all defect (Figure 5a), regardless of the number of processors.

In order to better understand why cooperative processors flip to being defective, we also plotted the corresponding weighted utility of processors in Figure 6. In case of both fair and incentive-compatible protocols, the average utility drops significantly with increasing number of processors. The average utility gradually converges at about $-c^m$ (which is -10 in our simulation). As utility by cooperation drops below $-c^m$, processors flip to being defective and incur only $-c^m$. Also, in uniform reward sharing we see a constant $-c^m$ utility for all (defective) processors.

E. DISCUSSION AND LIMITATIONS

Our goal in this work was to design practical incentive mechanisms for eliciting cooperation in shard-based blockchains. The above analytical and empirical results show how our proposed reward sharing mechanism promotes cooperation in shard-based blockchains, and thwarts free-riding processors. Nonetheless, there exists certain limitations in the proposed mechanism, discussed below, some of which we plan to address in the future.

1) INTER-SHARD COMMUNICATION

Due to the lack of communication between committees, cooperative processors in a shard where consensus is reached,

can suffer when another committee fails to reach consensus (because no block is added to the blockchain if one or more shards fail). This can be resolved if an inter-shard communication is established in Algorithm 1, wherein coordinators can exchange consensus status and inform potentially cooperative processors about the state of consensus in other shards as well. We plan to include inter-shard communication in our future work, and analyze how the game changes due to it.

2) INCLUSION OF MALICIOUS PROCESSORS

In this work we consider only honest but greedy (or selfish) processors (each trying to maximize its utility) who would follow the instructions of a coordinator. However in real-world, malicious processor(s) may also exist whose sole objective may be to disrupt the blockchain network. Such malicious processors may misbehave at various stages of the protocol, such as reporting false $H(x_i^j)$ or not following coordinator's instruction to cooperate (or defect). As part of our future work, we plan to include malicious processors in the game and re-analyze the game.

3) PARAMETRIC VALUES

The parametric values chosen for our numerical analysis were primarily to showcase the trends observable across the three different reward sharing mechanisms. They may or may not be reflective of values in a real shard-based blockchain network, but we did our best to establish the inequalities between parameters as completely as possible.

VI. RELATED WORK

In this section, we briefly outline the efforts in the literature towards improving the scalability and transaction rate of consensus protocols in public permissionless blockchains. For an exhaustive survey of blockchain consensus protocols in the literature, readers should refer to [18]. In the context of our work, the original *Nakamoto consensus* protocol [1] of *Bitcoin* which employed a leader selection using PoW puzzles (to commit the next block) suffers from poor scalability and transaction throughput. *Bitcoin-NG* [6] attempted to improve Bitcoin's performance by employing *microblocks*. In Bitcoin-NG, similar to Bitcoin, a leader is selected using PoW in each epoch. However, unlike Bitcoin, the leader can continue to append microblocks (containing transactions) to the blockchain for the duration of its epoch, until a new leader is elected.

As leader (or single processor) based implicit consensus algorithms such as Nakamoto consensus and Bitcoin-NG still suffer from poor performance, fault-tolerance and consistency issues, the community's focus shifted on designing blockchain consensus protocols using a *committee* of processors, rather than a single processor (or leader). While committee-based consensus algorithms were introduced more than two decades ago [19], much recently Decker *et al*. [20] proposed one of the first committee-based consensus protocols for public blockchains, named *PeerCensus*. However, PeerCensus did not clarify how committee formation is done and how an honest majority can be ensured within the committee. Follow up works [21]–[24] in similar direction improved the practicality of such single committee-based consensus protocols by proposing different strategies on how unbiased committees can be formed.

Although single committee consensus algorithms provide significantly improved performance compared to single processor or leader-based consensus algorithms, one major limitation of such techniques is that they do not scale well. Moreover, increasing committee size in such techniques comes at the expense of a decreased throughput. This motivated the design of blockchain consensus protocols that employ *multiple committees*. The main idea in these protocols is to split the transactions among multiple committees (or shards), which then process these shards or set of transactions in parallel. This also improves the overall scalability of the system. *RSCoin* [25] was proposed as a shard-based blockchain technique for centrally-banked cryptocurrencies, while *Elastico* [8] was the first shard-based consensus protocol for public blockchains. *Omniledger* [10] and *Rapidchain* [9] are some of the recently proposed shard-based public blockchain protocols that attempt to address the scalability and security issues of Elastico. Despite the recent interest in shard-based protocols for improving transaction throughput and scalability in public blockchains, there have been no prior efforts in the literature, until this one, that study the rational behavior of processors or miners in such a multiple committee approach.

VII. CONCLUSIONS

In this paper, we comprehensively studied the problem of selfishness in shard-based permissionless blockchains. We first introduced a system model to capture the main operational parameters in current shard-based blockchain protocols. Next, we evaluated the strategic behavior of processors in such protocols by employing concepts from game theory. Specifically, we modeled shard-based blockchain protocols as n-player non-cooperative games using different reward sharing scenarios and obtain the Nash equilibria (NE) strategy profile for each scenario. Based on our analytical results under different reward sharing scenarios, we designed an incentive mechanism for shard-based blockchain protocols which would enforce cooperation among processors by guaranteeing optimal incentive distribution. Our numerical analysis also validated that the proposed reward sharing mechanism outperforms uniform reward sharing and provides more incentive for cooperation when the block reward or number of transactions is small. This work is the first step towards a deeper understanding of the effects of non-cooperative behavior in shard-based blockchains.

ACKNOWLEDGMENT

This work was completed while Dr. Manshaei was visiting the University of Texas at San Antonio as a summer research fellow

REFERENCES

[1] S. Nakamoto. (2009). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: http://bitcoin.org
[2] (Jul. 2018). *All Cryptocurrencies*. [Online]. Available: https://coinmarketcap.com/all/views/all/
[3] (Jul. 2018). *Ethereum Project*. [Online]. Available: https://ethereum.org/
[4] (Jul. 2018). *Scalability—Bitcoin Wiki*. [Online]. Available: https://en.bitcoin.it/wiki/Scalability/
[5] J. Garzik. (2015). *Bitcoin Improvement Proposal 102*. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0102.mediawiki
[6] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *Proc. NSDI*, 2016, pp. 45–59.
[7] (Jul. 2018). *The Raiden Network*. [Online]. Available: https://raiden.network/
[8] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 17–30.
[9] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: A fast blockchain protocol via full sharding," Cryptol. ePrint Arch., Tech. Rep. 2018/460, 2018. [Online]. Available: https://eprint.iacr.org/2018/460
[10] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via sharding," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 19–34.
[11] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, 2002.
[12] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. OSDI*, vol. 99. 1999, pp. 173–186.
[13] (Jul. 2018). *Blockchain Explorer*. [Online]. Available: https://www.blockchain.com/explorer
[14] J. Nash, "Non-cooperative games," *Ann. Math.*, vol. 54, no. 2, pp. 286–295, 1951.
[15] A. Tucker, "A two-person dilemma," in *Readings in Games and Information*, E. Rasmusen, Ed. Oxford, U.K.: Blackwell, 2001, pp. 7–8.
[16] H. Hamburger, "N-person Prisoner's dilemma," *J. Math. Sociol.*, vol. 3, no. 1, pp. 27–48, 1973.
[17] M. Archetti and I. Scheuring, "Game theory of public goods in one-shot social dilemmas without assortment," *J. Theor. Biol.*, vol. 299, pp. 9–20, Apr. 2012.
[18] S. Bano *et al.* (Nov. 2017). "Consensus in the age of blockchains." [Online]. Available: https://arxiv.org/abs/1711.03936
[19] G. Bracha, "An O(log n) expected rounds randomized byzantine generals protocol," *J. ACM*, vol. 34, no. 4, pp. 910–920, 1987.
[20] C. Decker, J. Seidel, and R. Wattenhofer, "Bitcoin meets strong consistency," in *Proc. 17th Int. Conf. Distrib. Comput. Netw.*, 2016, Art. no. 13.
[21] I. Abraham, D. Malkhi, K. Nayak, L. Ren, and A. Spiegelman. (Dec. 2016). "Solida: A blockchain protocol based on reconfigurable Byzantine consensus." [Online]. Available: https://arxiv.org/abs/1612.02916
[22] R. Pass and E. Shi, "Hybrid consensus: Efficient consensus in the permissionless model," in *Proc. Int. Informat. LIPIcs-Leibniz*, 2017, pp. 39:1–39:16.
[23] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *Proc. 25th USENIX Secur. Symp. (USENIX Security)*, 2016, pp. 279–296.
[24] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proc. 26th Symp. Oper. Syst. Princ.*, 2017, pp. 51–68.
[25] G. Danezis and S. Meiklejohn. (May 2015). "Centrally banked cryptocurrencies." [Online]. Available: https://arxiv.org/abs/1505.06895

MOHAMMAD HOSSEIN MANSHAEI received the B.Sc. degree in electrical engineering and the M.Sc. degree in communication engineering from the Isfahan University of Technology, Iran, in 1997 and 2000, respectively, and the M.Sc. degree in computer science and the Ph.D. degree in computer science and distributed systems from the University of Nice, Sophia-Antipolis, France, in 2002 and 2005, respectively. He did his thesis work at INRIA, Sophia-Antipolis. From 2006 to 2011, he was a Senior Researcher and a Lecturer with the Swiss Federal Institute of Technology, Lausanne. He held visiting positions at UNCC, NYU, VTech, and UTSA. He is currently an Associate Professor with the Isfahan University of Technology. His research interests include wireless networking, wireless security and privacy, computational biology, and game theory.

MURTUZA JADLIWALA received the bachelor's degree in computer engineering from Mumbai University, India, and the Ph.D. degree in computer science from the State University of New York at Buffalo, USA. He was a Post-Doctoral Research Fellow with the Department of Computer and Communication Sciences, Swiss Federal Institute of Technology, Lausanne, from 2008 to 2011, and an Assistant Professor with the Department of Electrical Engineering and Computer Science, Wichita State University, USA, from 2012 to 2017. He also served as a Summer Faculty Fellow with the U.S. Air Force Research Laboratory—Information Institute, Rome, NY, USA, in 2015. He is currently an Assistant Professor with the Department of Computer Science, The University of Texas at San Antonio, USA. His current research is focused towards overcoming security and privacy threats in networked computer and cyber-physical systems.

ANINDYA MAITI received the M.S. degree in electrical engineering and the Ph.D. degree in electrical engineering and computer science from Wichita State University, USA. He is currently a Post-Doctoral Fellow with The University of Texas at San Antonio, USA. His current research is primarily focused towards uncovering and solving privacy and security problems in smart home and other IoT devices.

MAHDI FOOLADGAR received the B.Sc. degree in software engineering working on blockchain-based voting protocols from the Isfahan University of Technology, where he is currently pursuing the M.Sc. degree in software engineering with the Department of Electrical and Computer Engineering, under the supervision of Dr. Manshaei. His current research interests include blockchain, game theory, and security protocols.