Computing and Informatics, Vol. 40, 2021, 522–542, doi: 10.31577/cai_2021_3_522

A DECENTRALIZED AUTHORITATIVE MULTIPLAYER ARCHITECTURE FOR GAMES ON THE EDGE

Aleksandar Tošić

University of Primorska, The Andrej Marušič Institute
Muzejski Trg 2, 6000 Koper, Slovenia
&
Innorenew CoE
e-mail: aleksandar.tosic@upr.si

Jernej Vičič

University of Primorska, The Andrej Marušič Institute
Muzejski Trg 2, 6000 Koper, Slovenia
&
Research Centre of the Slovenian Academy of Sciences and Arts
The Fran Ramovš Institute
e-mail: jernej.vicic@upr.si

Abstract. With the ever growing number of edge devices, the idea of resource sharing systems is becoming more appealing. Multiplayer games are a growing area of interest due to the scalability issues of current client-server architectures. A paradigm shift from centralized to decentralized architectures that would allow greater scalability has gained a lot of interest within the industry and academic community. Research on peer to peer network protocols for multiplayer games was mainly focused on cheat detection. Previously proposed solutions address the cheat detection issues on a protocol level but do not provide a holistic solution for the architecture. Additionally, existing solutions introduce some level of centralization, which inherently introduces single point of failures. We propose a blockchain-based, completely decentralized architecture for edge devices with no single point of failure. Our solution relies on an innovative consensus mechanism based on verifiable delay functions that additionally allows the network to derive verifiable randomness.

We present simulation results that show the assignment of players and referees to instances is pseudo-random, which inherently prevents collusion-based cheats and vulnerabilities.

Keywords: Edge computing, consensus, peer to peer, network protocol, multiplayer games, blockchain

Mathematics Subject Classification 2010: 68T50

1 INTRODUCTION

The gaming industry is worth almost 135 billion at the time of writing [7]. The same source predicts a steady 10 % growth in the next 2 years, reaching 180 billion by the end of 2021. The recent trends toward multiplayer games have been very successful with games like Fortnight earning more than 2.4 billion in revenue in 2018 alone [26]. Steam, the biggest game distribution platform reported it serves as much as 18.5 million clients concurrently. Cloud computing enabled servers need to be migrated real time in order to meet the demand of clients. Additionally, network latency was reduced due to localisation approaches where servers are spawned geographically close to clients if possible. However, maintaining a player base of thousands or even millions together with the hardware and software infrastructure is both very expensive and difficult to maintain [29]. The recent idea of a "sharing economy" can be applied in tandem with the paradigm shift to edge computing. More specifically, clients on the edge of the system can profit from sharing resources, such as bandwidth and computing power, thereby releasing the burden on centralized servers.

This can be achieved by using a peer to peer (P2P) architecture. P2P gaming architectures have been studied extensively but have not been widely adopted [29]. The main issues are closely related to the lack of authority and trust. Centralized architectures solve these issues with authoritative servers. The server's tasks are to simulate game play, validate and resolve conflict in the simulation, and store the game state. P2P multiplayer architectures were previously able to address some of the cheating vectors but required some level of centralization.

More recently, blockchain technology has gained a significantly large interest. Research in cryptography and fault tolerant consensus mechanisms has been driving the evolution of decentralized P2P systems.

The already available schemes that, at least theoretically, address most of the identified cheats in distributed gaming architectures RACS [28] and Goodman [12] still retain a central authority either to store the game state or as a refereeing authority and, thus, still retain the Single Point Of Failure – (SPOF) [8] property. Our research presented in this paper mostly focuses in the elimination of the SPOF but still being able to successfully address the same set of cheats. We were able to

achieve the set goals and were even able to partially address the Collusion cheat, as described in Section 5.7.

2 STATE OF THE ART

Baughman et al. [3] propose an improvement of their lock-step protocol [2] Asynchronous Synchronization (AS), the first protocol for providing cheat-proof and fair play-out of centralized and distributed network games. The protocol also provides implicit robustness in the face of packet loss. At proving the correctness of their approach, they make a number of assumptions: there exists a reliable channel between all players; all players know of all other players; players are able to authenticate messages from each other player; and all players wait only a finite time before making decisions and revealing commitments. Their approach can be implemented in a true peer-to-peer fashion, thus eliminating the SPOF, but, as it can be seen from Table 1, the approach is not immune to Replay/Spoof cheat [28].

GauthierDickey et al. [11] present a protocol designed to improve on lock-step protocol [2] by reducing latency while continuing to prevent cheating. They achieve this by adding a voting mechanism to compensate for packet loss in the environment. They call this protocol New Event Ordering (NEO).

Corman et al. [6] present SEA protocol and argue that it outperforms NEO algorithm in all cheat prevention properties; further, they present three possible cheats that the NEO protocol fails to address: Attacker can replay updates for another player. Attacker can construct messages with any previously seen votes attached. Since the votes are signed, the messages will appear to come from another player. Attacker can send different updates to different opponents.

Cronin et al. [14] present SP protocol which addresses the late-commit cheat and presents a performance improvement on the existing protocols (lock-step).

Goodman [12] proposes IRS hybrid C/S – P2P design; it operates by routing request messages through a centralized server and relaying them to proxy clients, a secure method by which it is certain that the requesting and proxy clients received the same message. The proxy clients perform calculations for others, relieving the server of the calculation burden. The code of the IRS approach relies on identifying malicious clients. This can be done with a certain probability and can still lead to cheat exploits.

Pellegrino and Dovrolis [20] propose a change from Client-Server architecture to Peer-to-Peer with Central Arbiter architecture (PP-CA) that contains server bandwidth requirements when increasing number of players, effectively solving the biggest scalability problem. The system still retains the SPOF in the form of the centralized arbiter. The paper focuses entirely on the elimination of the bandwidth problem and does not deal with any cheats; actually, it introduces a new form of cheating (e.g., blind opponent – BO, discussed in Section 3).

Webb et al. [28] compare all algorithms and show that all the previous distributed protocols and schemes are vulnerable to several cheats. Their proposed

scheme (RACS), which extends PP-CA [20], solves most of the problems but still has the SPOF in the form of the Identity server and Referee: a process running on a trusted host that has authority over the game state.

Most of the presented protocols are SPOF-free as they address only the P2P communication protocol but are vulnerable to cheats, as can be seen on Table 1. The RACS and IRS address most of the cheats but reintroduce the SPOF. Our scheme eliminates the SPOF problem and still retains all the properties described in RACS [28] and at least partially deals with the Collusion cheat that, at least in our opinion, cannot be eliminated by means of protocols and technology.

3 CHEAT TAXONOMY

In this article, we use the definition of Yan and Randel [30] for online game cheating: "Any behaviour that a player uses to gain an advantage over his peer players or achieve a target in an online game is cheating if, according to the game rules or at the discretion of the game operator (i.e., the game service provider, who is not necessarily the developer of the game), the advantage or the target is one that he is not supposed to have achieved." Cheaters try to gain unfair advantage over other players. This can totally destroy the in-game economics of an online game or simply ruin the gaming experience. Grievers, as the name implies, are players with the sole intention of hurting other players' experience as much as possible. When this behaviour adheres to game rules, it is technically not a form of cheating and is out of scope of this paper. Both groups exploit the same set of cheats.

The taxonomy presented in Table 1 and accompanying text and later used in this paper follows the taxonomy presented in Web et al. [28] and Yahyavi et al. [29], we added 3 additional entities, 2 were not addressed by the previous research (Robustness and User data privacy), the last one (Lack of Devices Situation) is a consequence of our approach and not applicable to other architectures. It is addressed later in this section. Multiple authors addressed the issue of systematic classification of cheating in online games, such as Yan and Randel [30] who present a taxonomy of 15 types of cheats and just by comparing the number of entries we could assume that the later introduces additional forms. We argue that the set of cheats presented in our paper fully covers the whole set presented in Yan and Randel [30] with the addition of two new entries that are discussed separately. The translations are presented in Table 1 in the second column. The "Cheating by compromising passwords" can be classified as "Social engineering" class and as such omitted. We argue that these two entries cannot be successfully addressed by the game architecture, they must be addressed mostly by informing the players.

1. *Bug* – bugs in games can lead to potential misuse by the players. No scheme directly addresses this problem, it is assumed that the bugs will be fixed by software developers.
2. *IE, IC* – the goal of IE (Information Exposure) is to obtain secret information to which the cheater is not entitled, thus gaining an unfair advantage in selecting

	Yan	Problem/Cheat	RACS	IRS	C/S	AS	NEO	Damage
1	L	Bug	√	√	√	√	√	√
2	A, F, H	IE, IC	√	√	√	×	×	√
3		Bots	×	×	×	×	×	×
4	A, C, F	Supp. update, TS, FD	√	√	√	√	√	√
5	A, F, H	Replay, Spoofing	√	√	√	×	√	√
6	A, D, F, H	Undo	n/a	n/a	n/a	√	×	n/a
7	A, C, F, H	BO	√	√	n/a	n/a	n/a	√
8	G	DDoS	√	√	√	√	√	√
9	B	Collusion	×	×	×	×	×	$\surd^{1}$
10	M,	Robustness	×	×	×	×	×	√
11	J	User data privacy	√	√	√	×	×	√
12		Lack of Devices Situation	n/a	n/a	n/a	n/a	n/a	√
13	E	Exploiting AI	n/a	n/a	n/a	n/a	n/a	n/a
14	I, O	Social Engineering	n/a	n/a	n/a	n/a	n/a	n/a^{*}

Table 1. Chart presenting all identified cheats and schemes for detection and removal. The n/a is given to a scheme that does not have to deal with the observed cheat for implicit reasons (mostly architectural).

the optimal action. The IC (Invalid Command) cheat occurs when an application or data files are modified to issue commands or command parameters that originally could not be generated.

3. *Bots* – programs that act as players can be introduced to the game. The programs can exercise number of cheats including Collusion.
4. *Supp. update, TS, FD* – A cheater can suppress sending the state update, send the updates at a slower rate (FD) to gain advantage or incorrectly timestamp messages to gain advantage.
5. *Replay, Spoofing* – a player can obtain advantage by replicating messages by means of local software instead of using the tools provided by the game (example: sending impossibly fast series of missiles).
6. *Undo* – a player succeeds to undo a previously sent message after already receiving the opponents message and realizing that the original message was not optimal.
7. *BO* – in distributed schemes that use a Central Arbiter (CA), such as PP-CA [20], cheater may purposely withhold updates to his peers (but not to the CA), effectively covering own actions.
8. *DDoS* – a (cheating) player may use DDoS [17] attack to temporally disable the opponent to send messages and thus get advantage.
9. *Collusion* – unfavorable situation may occur whereby certain clients cooperate with one another in order to gain unfair advantage over others. Collusion via the use of external communication is difficult to eliminate due to the use of non-monitored means of communication [12].

10. *Robustness* – The robustness metric shows how much is the system or scheme or protocol fault tolerant (how much it is tolerant to node failure, at the worst to server node failure). The basic Client-Server architecture is the least robust and totally decentralized system is the most robust.
11. *User Data Privacy* – user profiles with scoring and possibly in-game funds and purchases are stored for future use. Client Server architectures use the server as a means for reliable storage, distributed systems have to deal with security risks as the data is spread on the network or stored locally at clients an thus easily available for tempering.
12. *Lack of Devices Situation* – all schemes that rely on an outer referee (an external entity that is not part of the game) rely on the availability of the referee. In decentralised architectures where refereeing is done by other players, the scheme relies on availability of adequate number of players.
13. *Exploiting AI* – Exploiting artificial intelligence (AI) cannot be handled within the protocol or the architecture. The idea that a player can use an AI to improve its decision making in a game is not possible to detect on the protocol level.
14. *Social Engineering** – social engineering is a very broad term. Generally, it involves using social information about a player to trick the person into revealing sensitive information pertaining to a game, i.e. passwords. Our protocol uses ECDSA public cryptography, and does not require passwords. The authentication is not necessary as messages exchanged between players are all signed and verified.

4 DECENTRALIZED ARCHITECTURE FOR THE EDGE

Previously proposed P2P architectures rely on some level of centralization. We propose a completely decentralized architecture for edge devices that would inherently circumvent the single point of failure (SPOF). In contrast to client-server (C/S) architectures, where the server is authoritative, P2P networks are arguably more exposed to cheats and vulnerabilities. To address the issues Web et al. propose RACS [28], a referee node that takes the authoritative role in case of conflicting or inconsistent states between players.

However, RACS does not address issues of node selection. In completely decentralized networks, deriving secure randomness is an open question. From a security point of view, a decentralized random generator must not be known in advance to avoid attacks and vulnerabilities based on information exposure (IE). At the time of writing, most networks rely on oracle networks secured with game theoretical incentive schemes [21, 10]. However, the security models for such systems require strong incentives, which are mostly based on staking mechanisms that introduce penalties for bad actors and rewards for good actors [13]. A recent paper proposed a mathematical construction for verifiable random functions (VDF) [4], an extension of time lock puzzles [22] that produce verifiable proofs of computation. More specifically,

VDFs are similar to time lock puzzles but require a trusted setup where the verifier prepares each puzzle using its private key. Additionally, a difficulty parameter can be adjusted to increase the amount of sequential work, thereby increasing the delay. The proof can be used as an entropy pool for a seeded random to derive randomness within a decentralized system while preventing attacks based in IE. We solve the requirement for a trusted set up (private key of the VDF) by using a blockchain structure. Blocks have a configurable block time parameter, which is used to adjust the difficulty parameter of the VDF, to target the block time. The consensus algorithm is a novel lottery draw scheme, where nodes draw lottery tickets in order to be voted as block producers.

Suppose the current block is H at height (canonical id) h. The block hash of block H is used to compute the VDF and obtaining proof H_p. Each node $n \in N$ then draws its own lottery ticket H_t, which is defined as the distance between the node's public key, and H_p. Since all nodes share the same H_p, and all nodes (asymptotically) computed H_p at the "same" time, the lottery draw is not predictable. A node is elected to be part of the validator set if H_t is within the v closest tickets, where v is a configurable parameter usually set to $\frac{P}{PPI}$, where P is the total number of players, and PPI is the number of players per instance. Nodes that belong to the validator set are considered referees, and block producers for block $H + 1$. The structure of the block is shown in Figure 1.

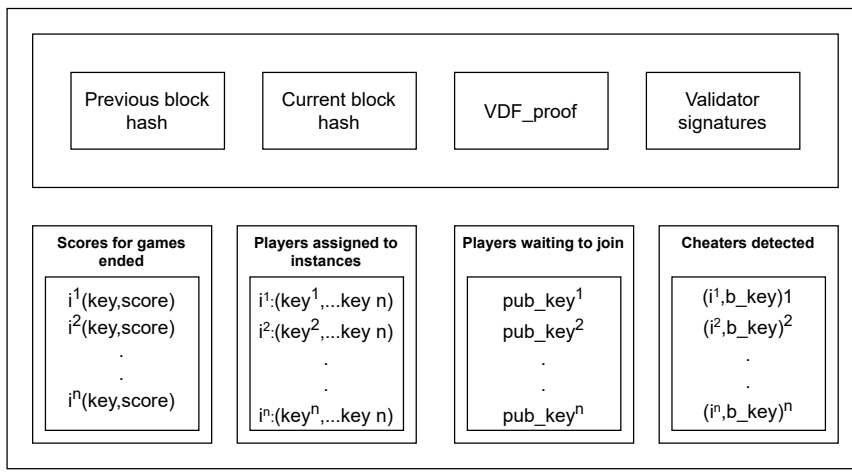

Figure 1. The Header of a block contains the previous block hash, current block hash, VDF proof, and signatures of all validators that signed the block. The body of the block contains a list of game instances that completed (combination of players public addresses), and their scores, a list of players waiting to join the next round (block), and a list of instances with assigned players that were in the waiting queue of the previous block.

Players are uniquely identified by their public key. Each player generates a public-private key pair before connecting to the P2P network. Upon joining the network,

players synchronize the last block to find the validator set. Querying the Distributed Hash Table – DHT [25], a node connects to the one or more validators to broadcast their intent to join a game. Once consensus is reached amongst the validators, a new block H will be forged that will include the player's public key in the players awaiting list. Players will receive block H, learn about their inclusion to the awaiting list, and wait for block $H + 1$. Note that the target block time is configurable with the VDF difficulty and should be set by the game operator. Upon receiving block $H + 1$, the players' public key will be assigned to an instance. Each instance has a unique ID, which is obtained by concatenating the public keys of players assigned to the instance. Each player parses the instance ID to obtain the public keys of opponents and connects to them by querying the DHT (with the public key) for their address. The last address of the instance ID is the assigned validator that will assume the role of the referee. Once the instance is resolved (game finished), the referee (also a validator) will inform all other validators and propose the inclusion of the decision/score in the next block. Validators will vote on the proposed block by signing it with their public key. Clients can verify their signature using their public key. In case a referee of an instance detected a cheater, a proof can be sent to the set of validators for confirmation. The details of how this is achieved are explained in detail in Section 5.

A candidate block $H+1$ is then transmitted (using gossip protocol) [15] through the P2P network. Each client accepts the block if and only if the block references the local block hash at height h, the provided proof H_p is valid, and the candidate block contains signatures of all validators whose public keys can be computed by each node using H_p. The nodes that are part of the validator set execute a matchmaking algorithm that must be deterministic (but can rely on randomness derived from H_p) and can use the previous block as input (list of players wanting to join). The matchmaking algorithm assigns player to game instances, and referees from the validator are set to act as authoritative nodes. The deterministic nature of the matchmaking algorithm is used to reach consensus amongst the validator nodes. The consensus is reached as all honest nodes will construct the same candidate block and will sign all candidate blocks equal to theirs. The result will be a candidate block signed by the majority of validators.

The construction assumes validators, referees, and players to be players. However, a set T of trusted nodes is required and assumed to be maintained by the game maintainers. These nodes are called full-nodes and are necessary to guarantee liveliness of the system even in extreme cases where there are no players in the network. Full-nodes are also responsible for permanently storing the blockchain and maintaining a DHT-based structure other nodes can query to discover other peers. Players are assumed to be lite clients that do not need to store the entire blockchain history in order to participate in the consensus [27]. Additionally, referees are assumed to be players as well. The matchmaking algorithm should avoid assigning players to be referees to their own game instance.

Each game can have one referee, which arguably decreases the robustness. All decisions about conflicts proposed by a referee must be presented to the validator

set in order to reach consensus and gather enough signatures to make the block valid. However, the referee can unexpectedly disconnect or even worse, be attacked by a player during the game. To circumvent this issue, any number of validators can be assigned as backups in case the referee is unresponsive.

Referees in DAMAGE are running the same protocol as RACS. However, in case a referee detects a cheating player, the proof (usually a set of states that allow validators to recreate/simulate the game) must be presented to the set of validators (also RACS referees). Consensus is reached if and only if $\frac{2}{3}$ validators agree [5]. Decisions about the proposed cheat detected is done by voting for the block. Each block contains a list of (*public key*, *instance key*) pairs and the type of cheat detected. Assuming the validator is honest, and the referee proposing the detected cheat is as well, both validators will reach the same conclusion and thus sign the block. In any other case, the block will only be signed by the malicious validator. Proposals that do not reach consensus are considered invalid blocks and will be rejected by the client protocol. A subset of nodes (without the majority vote) running modified clients may choose to accept the invalid block, thereby forking the chain [1]. In such cases, the next block would either resolve the fork if it is accidental or disconnect (network level) the subset of nodes with the modified protocol due to an invalid VDF proof on the forked chain.

5 SECURITY MODEL

All communication between peers provides the same level of security to that of C/S architectures by using public key (ECDSA) cryptography. DAMAGE provides a secure and completely decentralized protocol for selecting referees, and match players to game instances. However, the player and referee protocols are based on RACS [28] and therefore DAMAGE inherits cheat detection properties of RACS, and extends them with efficient and secure peer selection, peer synchronization, robustness, and some aspects of collusion.

5.1 Referee Selection

We address the issue of Referee selection by using VDF to derive randomness with which a lottery-based consensus is reached. The sequential nature of VDFs prevents IE attacks where a player would compute the VDF and, using the proof, obtain information about which nodes are part of the validator set and which node is assigned as the main referee to each game. Game operators should set the difficulty parameter according to their desired performance/security ratio. A more difficult VDF will result in players waiting to be matched to an instance longer (i.e., a few seconds), while a lower difficulty will potentially allow malicious players to discover the nodes that will be within the set of validators before others. We argue that knowing the set of validators and, consequently, the referee node for a game does not give the player a competitive advantage. This is further explained in the case of collusion.

5.2 Referee Trust

In RACS scheme referee nodes are assumed trustworthy (the authors acknowledge this to be an open issue). We solve this issue with the validator set. Even if a RACS-based referee is compromised, any player can dispute the referee and seek a decision by consensus within the set of validators. The player would then have to compromise $\frac{2}{3}$ nodes in the validator set, which is not known in advance and changed every block.

Instead of using only one referee per game, we propose to establish a Referee set (validator set of referees) that. Additionally, the cardinality of the validator set is a configurable parameter analogous to the trust level required by the game (higher trust requires bigger cardinality).

5.3 Synchronization

On the data layer, nodes synchronize through the blockchain. Blocks store the current state of the system on lite clients and the entire history on full-nodes maintained by the game operators. Blocks are gossiped across the network efficiently by maintaining a DHT that maps nodes (public keys) to their network addresses. Additionally, referees in the validator set must synchronize and reach consensus about the detected cheats and results of the games played. Due to the deterministic nature of the cheat detection algorithms, honest nodes will reach the same decision as the referee that reported the cheat. Consensus is reached if the majority of the validators sign the proposed block (which includes the decision about reported cheats).

5.4 Robustness

DAMAGE uses redundancy to increase fault tolerance. There are two main types of faults that can occur. A peer can fail (disconnect or violate protocol) before, after or during playing the game, and a peer acting as a referee (and also as a player in a different game) fails at the same time.

Player faults after entering matchmaking: A peer that faults after it announced inclusion to the validator set will cause the validators to match its public address to an instance. Other peers attempting to connect will fail and/or result in protocol violation. It is up to the client protocol to decide if the game instance can continue to run without the faulty peer or not. In case the instance must be destroyed, this can be trivially solved by extending the referee's protocol to label this as a "cheat". The referee will announce the instance destruction to the validator set.

Player faults during the game: If possible, the game instance should keep running. If not possible, the referee should notify the validator set about the destruction.

Player faults after the game: No effect on the system.

Faulty validators (referees) are arguably a bigger security issue. Even without the ability for players to know which referee will be assigned to their instance there is still a possibility of DDoS attacks on referees during the game. To combat this issue, validators form a randomly shuffled priority queue using the VDF proof. The priority queue is a backup queue of referees that will take over an instance in case the referee assigned faults. The fault tolerance can be increased on demand by increasing the size of the validator set. However, detecting a faulty referee must be done by peers playing in the instance. If messages from peer to referee are either latent or connection is dropped, client protocol will take the following steps:

1. Set up a seeded random with the latest block (local) of the VDF proof.
2. Compute the lottery draw results to find the public keys of the validator set.
3. Shuffle the validator set list with the same seed.
4. Contact the next validator (backup referee for its game).

5.5 User Profile Management

Previous research relied on a central authority for authenticating users and managing their profiles such as avatars, variables, and metadata. Our architecture can be extended with a completely decentralized storage and authentication service, a centralized authoritative server as well as inter-operability between both. A blockchain-based authentication service can be built in by extending the block structure [18]. Additionally, blocks can be used for persistent immutable storage. However, storing data in blocks raises scalability issues [31, 24] as the blockchain becomes hard to maintain even for full nodes. Hybrid approaches have been proposed where data is stored centrally whereas the signatures are stored on-chain [31]. This creates a tamper-proof system where data can be verified and trusted as any attempt to tamper with the data would invalidate the signature (hash) [24].

5.6 Lack of Devices Situation

The refereeing process relies on a set of validators that are randomly chosen for each block time-cycle. The randomness of selection ensures that a player cannot know who is refereeing the next game. Player's devices are used to act as validators. The pool of validators cannot be constructed if there are not enough players. It is developers' or game operator's task to supply enough (a fixed number that does not grow with player-base) resident secure services (servers) that act as starting validators. These actors also maintain the blockchain (full nodes).

5.7 Collusion

Assuming the game runs multiple instances, we argue collusion between players is not possible. We assume colluding players know and, hence, trust each other.

Figure 5 shows a graph of a simulation of 200 players as nodes. The edges represent the number of games (weight) a player was assigned another player as the referee for the instance the player was matched to. Simulating 1 000 blocks, the average degree was 200, and the graph density was 1. We observe that the assignment of a referee and opponents derived using the VDF proof are thereby random. Hence, players cannot know in advance which instance the set of validators will assign them to nor the referee that will observe the game. Section 6.2 and Figure 5 present an empirical evaluation of the "fairness" of the selection method based on VDF. Suppose the colluding players are able to compute the VDF proof faster then other players, and, hence, learn about the game instance assignment in advance. However, since the seed for the next VDF is the block hash, the colluding players can see at most one block time into the future. Every player must announce the desire to be matched to a game in the current block (players awaiting list). Matching awaiting players will be executed in the next block. Despite the ability to see one block in the future, colluding players seek assignment to the same instance since they must announce their willingness to play before they learn about the instance assignment even in the worst case scenario.

6 EVALUATION

The paper presents a scheme to eliminate all known cheats in a fully distributed game setting. The scheme eliminates the SPOF problem in previously presented hybrid P2P – Referee settings for solving game cheats. We base our solution on already presented solutions, mostly RACS [28]. We present simulation results leading to the following conclusions:

- The VDF based selection of referees and players is fair.
- The block propagation scales well.
- Block propagation times are acceptable for fast match making, and conflict resolution.
- Dynamic block size does not impact performance of the system.
- Players do not need to maintain a large number of outgoing connections.
- Latency and bandwidth do not substantially slow down information propagation through the network.

The scalability of the solution is addressed in two ways. The first is the scalability of the consensus mechanism and the ability to propagate state and state transition depending on the block size. In this test we show that the solution can scale to hundreds of thousands of nodes and achieve consensus.

The second scalability test is performed by introducing variance in latency and bandwidth to mimic the instability of home internet connections under standard TCP/IP parameters.

Since every player is also a node, and potentially a referee, we argue that scalability in terms of number of players can be derived by the aforementioned tests. Additionally, due to the nature of the P2P architecture, once players are matched into a game instance, the entire communication is done solely between them, and the referee of that instance, which is completely independent of the rest of the network, and hence does not impact the scalability.

In order to evaluate the solution a simulation environment was developed. We simulate a P2P network where each peer (player) has the following constraints:

- Local bandwidth constraints. Bandwidth constraints are assigned to peers joining the network based on the distribution obtained from the European report on network bandwidth [9].
- Maximal number of outgoing connections (out edge degree) is assigned to nodes (MAE), we ran tests with different values of this parameter, they are color-coded in Figure 3.
- Each node's connection is single-directional taking into account upload and download bandwidth constraints of sender and receiver.
- Each new connection is assigned a round trip time (RTT) to represent variance in latency. RTT values are assigned randomly fitting a Gaussian distribution on an interval $[30, 250]$ ms, the values were taken from the European report on network bandwidth [9].
- Actual throughput of each connection is estimated using the Mathis metric [16] with following parameters that were taken from real-life situations: maximum segment size (MSS) of 1460 bytes (most used in today's communications as shown in papers such as [23]), the connection's RTT, and a TCP packet loss probability of $p = 1.0 * 10^{-5}$ [19].

Nodes (players) join the network by connecting to one of the trusted nodes. Trusted nodes are those operated by the game maintainer and serve only as the entry point for new peers to discover other peers or if needed to persistent storage for player accounts. A node proceeds to run the peer discovery protocol building the DHT. When new nodes are discovered, the peer attempts to make new connections until the MAE limit is reached and the node is considered to be well connected. Examples of different architectures (presented by connected directed graph) for 20 nodes are presented in Figure 2.

Once a new block is forged the origin nodes propagates the block using a basic flooding algorithm simulating the bandwidth, and TCP constraints. In each simulation, multiple directed graphs are constructed following the above protocol. Simulations were carried out with different number of nodes to observe the scalability of the solution. We measure propagation time as the total time it takes for all nodes to receive a newly forged block.

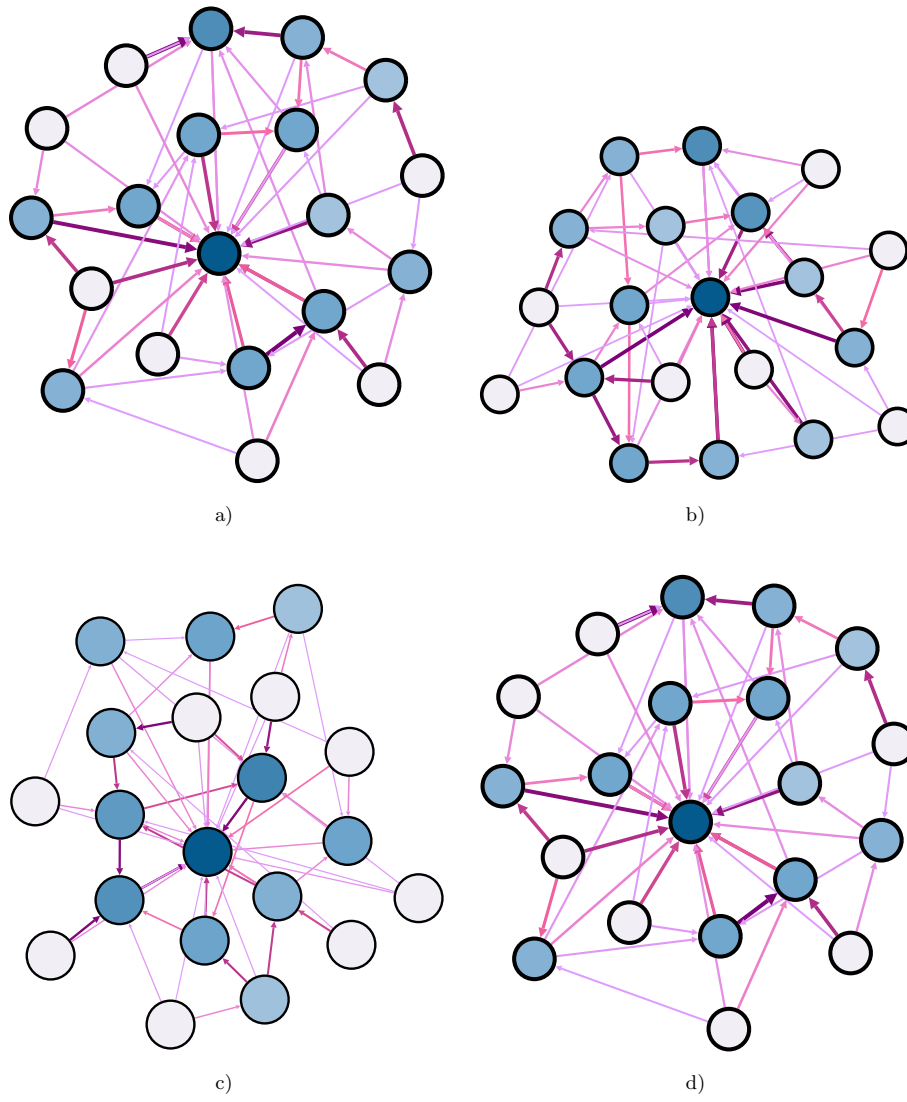

Figure 2. Examples of different architectures obtained by simulation. Number of nodes in all examples is $n = 20$.

6.1 Block Propagation Times

Blocks hold the state of the match making and games being played. Lowering the block time (VDF difficulty) would result into a more responsive experience. However, lower block times reduce security, and can cause network congestion. To avoid possible client synchronization issues the network must be able to reliably propagate blocks before new ones are forged. Additionally, the propagation times vary depending on the network topology, block sizes, and average node degree. We evaluate the scalability of propagating blocks in order to estimate viable block times, and show the scalability of the solution. From Figure 3 we observe that propagation times scale logarithmic as we increase the number of clients. Additionally, increasing the number of outgoing connection a node maintains reduces the average propagation times as it reduces the risks of unfavorable graph typologies.

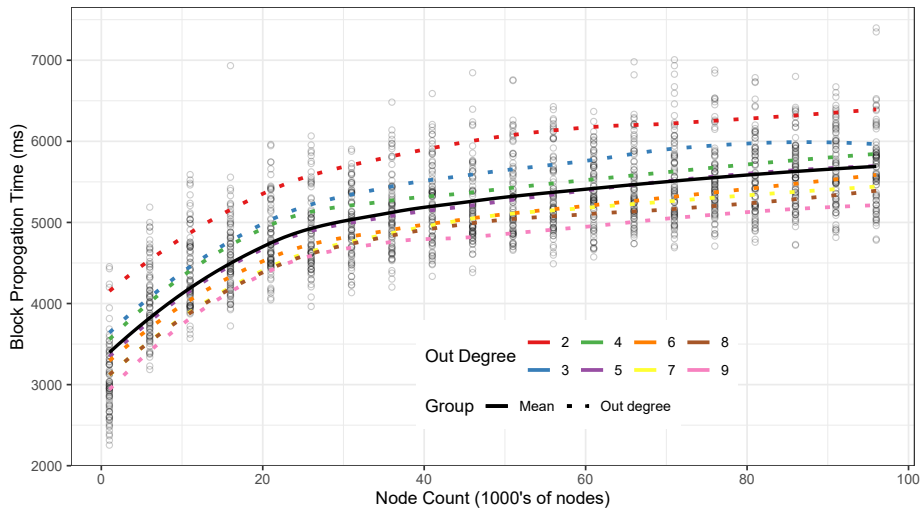

Figure 3. Simulation of block propagation on different graph configurations. Configurations are obtained by increasing the node count (number of players), and out degree using a block size of 1 MB.

Block size scales lineally with the number of clients. Every block has a constant size for the header, which is 64 bytes for previous and current block hash, 100 bytes for the VDF proof, and at least one validator signature of 64 bytes. As more players join the network, more games need to matched, assigned to instances, and scores saved. Figure 4 shows how the network scales with different block sizes. We observe that latency and bandwidth speeds of some nodes can cause considerable propagation slowdowns indicated by some outliers. However, this can be mitigated by having nodes maintain a dynamic number of outgoing connections increasing the

limit as blocks become larger (more players), and lowering the limit as blocks are smaller.

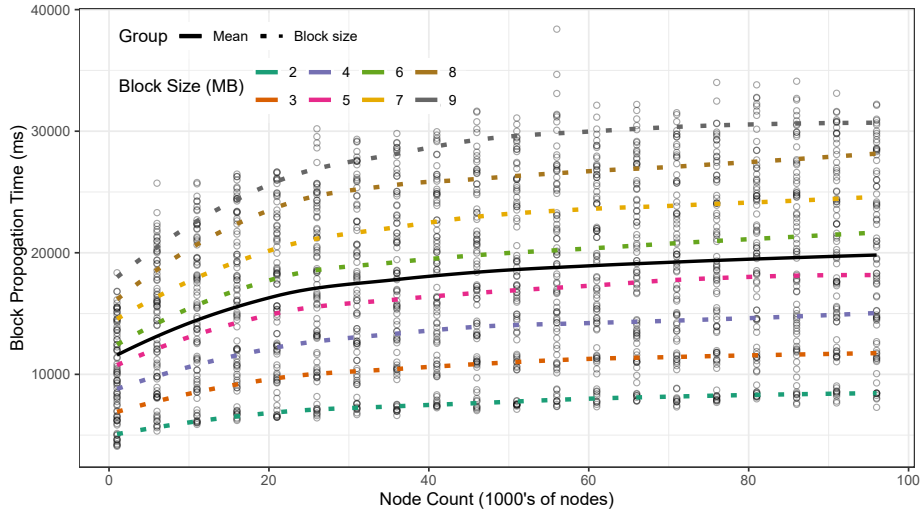

Figure 4. Simulation of block propagation on different graph configurations. Configurations are obtained by increasing the block size. Nodes were limited to 3 outgoing connections.

6.2 The VDF Based Selection of Referees and Players is Fair

Figure 5 shows a graph of a simulation of 200 players as nodes. The edges represent the number of games (weight) a player was assigned another player as the referee for the instance the player was matched to. Simulating 1 000 blocks, the average degree was 200, and the graph density was 1. We observe that the assignment of a referee and opponents derived using the VDF proof are thereby random. Hence, players cannot know in advance which instance the set of validators will assign them to nor the referee that will observe the game.

6.3 The VDF Method Scales Well

The setting presented in Section 6.2 and Figure 5 shows that the VDF based selection method is fair, the graph shows 200 players and 1 000 blocks, as this was the maximum feasible number combination that was still manageable to visualize. The setting was further evaluated with different parameters for the number of players, number of players per game and number of blocks. The number n of players per game: means a game where n players participate, Number of blocks: how much time the matchmaking process was observed. We evaluated the setting with 200, 1 000

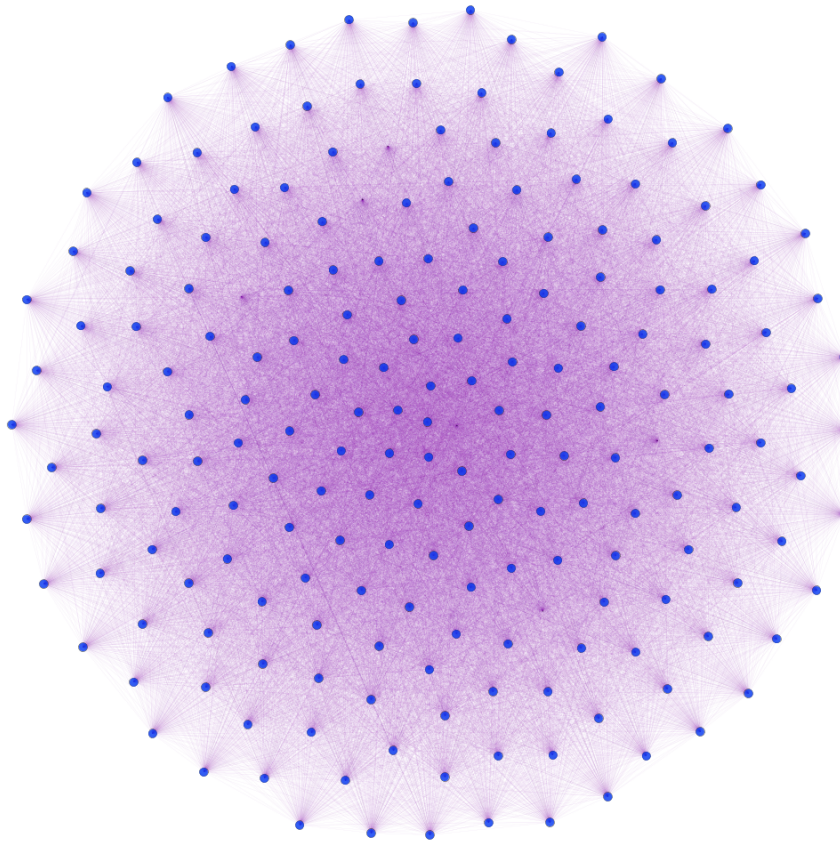

Figure 5. Simulation of 200 players in a 2-player game ($PPI = 2$) that played a total of 1 000 games. Nodes are players and edges represent instances where the destination node was referee for the instance the player was matched to.

and 10 000 players, 1 000, 2 000 and 10 000 blocks, we also changed the number of players per game. All results were consistent with the first test, the degree of all players was near the number of players and the density of the graph was near 1.

7 CONCLUSION AND FURTHER WORK

The paper proposes a blockchain-based, completely decentralized architecture for edge devices with no single point of failure that successfully addresses cheat problems. The presented solution is based on two hybrid approaches to P2P network games anti-cheat schemes that were based of server acting as referees. We propose

a completely decentralised approach while still retaining the same cheat resistance, actually in the case of Collusion we were able to partially address the issue. The proposed solution has not been fully implemented, we implemented the newly proposed building stones and executed empirical testing on a pilot setting. As the solution addresses the cheating problem in all aspects, a fully functional implementation is possible. DAMAGE is applicable to most game types. However, it is most suitable for turn based games, where potential latency does not impact user experience dramatically. Additionally, it reduces the complexity of the referee implementation due to the simple ordering of actions in the discrete time. Our results show, that the architecture scales automatically with the number of players thereby drastically reducing operation costs of running a multiplayer game. Every player added to the system also becomes a node, sharing its resources and contributing to verification as a potential referee.

Acknowledgment

The authors gratefully acknowledge the European Commission for funding the InnoRenew CoE project (Grant Agreement No. 739574) under the Horizon2020 Widespread-Teaming program and the Republic of Slovenia (Investment funding of the Republic of Slovenia and the European Union of the European Regional Development Fund).

REFERENCES

[1] Baliga, A.: Understanding Blockchain Consensus Models. Technical Report, Persistent Systems Ltd., 2017, pp. 1–14.

[2] Baughman, N. E.—Levine, B. N.: Cheat-Proof Playout for Centralized and Distributed Online Games. Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM 2001), Vol. 1, 2001, pp. 104–113, doi: 10.1109/INFCOM.2001.916692.

[3] Baughman, N. E.—Liberatore, M.—Levine, B. N.: Cheat-Proof Playout for Centralized and Peer-to-Peer Gaming. IEEE/ACM Transactions on Networking (ToN), Vol. 15, 2007, No. 1, pp. 1–13, doi: 10.1109/TNET.2006.886289.

[4] Boneh, D.—Bonneau, J.—Bünz, B.—Fisch, B.: Verifiable Delay Functions. In: Shacham, H., Boldyreva, A. (Eds.): Advances in Cryptology – CRYPTO 2018. Springer, Cham, Lecture Notes in Computer Science, Vol. 10991, 2018, pp. 757–788, doi: 10.1007/978-3-319-96884-1_25.

[5] Castro, M.—Liskov, B.: Practical Byzantine Fault Tolerance. Proceedings of the Third Symposium on Operating Systems Design and Implementation, New Orleans, USA, 1999, pp. 173–186.

[6] Corman, A. B.—Douglas, S.—Schachte, P.—Teague, V.: A Secure Event Agreement (SEA) Protocol for Peer-to-Peer Games. First International Confer-

ence on Availability, Reliability and Security (ARES '06), 2006, 8 pp., doi: 10.1109/ARES.2006.15.

[7] Dobrilova, T.: How Much is the Gaming Industry Worth? Techjury, 2019.

[8] Dooley, K.: Designing Large Scale LANs: Help for Network Designers. O'Reilly Media, Inc., 2001, 404 pp.

[9] European Court of Auditors: Broadband in the EU Member States: Despite Progress, Not All the Europe 2020 Targets Will Be Met. Technical Report, European Court of Auditors, 2018.

[10] Gatteschi, V.—Lamberti, F.—Demartini, C.—Pranteda, C.—Santamaría, V.: Blockchain and Smart Contracts for Insurance: Is the Technology Mature Enough? Future Internet, Vol. 10, 2018, No. 2, Art. No. 20, 16 pp., doi: 10.3390/fi10020020.

[11] GauthierDickey, C.—Zappala, D.—Lo, V.—Marr, J.: Low Latency and Cheat-Proof Event Ordering for Peer-to-Peer Games. Proceedings of the 14th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '04), 2004, pp. 134–139, doi: 10.1145/1005847.1005877.

[12] Goodman, J.: A Hybrid Design for Cheat Detection in Massively Multiplayer Online Games. M.Sc. Thesis, McGill University, Montréal, 2008.

[13] Heiss, J.—Eberhardt, J.—Tai, S.: From Oracles to Trustworthy Data On-Chaining Systems. Proceedings of IEEE International Conference on Blockchain (Blockchain 2019), 2019, pp. 496–503, doi: 10.1109/Blockchain.2019.00075.

[14] Jamin, S.—Cronin, E.—Filstrup, B.: Cheat-Proofing Dead Reckoned Multiplayer Games. Proceedings of 2nd International Conference on Application and Development of Computer Games, Hong Kong, 2003, pp. 1–7.

[15] Kwiatkowska, M.—Norman, G.—Parker, D.: Analysis of a Gossip Protocol in PRISM. ACM SIGMETRICS Performance Evaluation Review, Vol. 36, 2008, No. 3, pp. 17–22, doi: 10.1145/1481506.1481511.

[16] Mathis, M.—Semke, J.—Mahdavi, J.—Ott, T.: The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. ACM SIGCOMM Computer Communication Review, Vol. 27, 1997, No. 3, pp. 67–82, doi: 10.1145/263932.264023.

[17] Mirkovic, J.—Reiher, P.: A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. ACM SIGCOMM Computer Communication Review, Vol. 34, 2004, No. 2, pp. 39–53, doi: 10.1145/997150.997156.

[18] Moinet, A.—Darties, B.—Baril, J.-L.: Blockchain Based Trust and Authentication for Decentralized Sensor Networks. 2017, pp. 1–6, arXiv: 1706.01730, doi: 10.1109/wimob.2017.8115791.

[19] Moltchanov, D.: A Study of TCP Performance in Wireless Environment Using Fixed-Point Approximation. Computer Networks, Vol. 56, 2012, No. 4, pp. 1263–1285, doi: 10.1016/j.comnet.2011.11.012.

[20] Pellegrino, J. D.—Dovrolis, C.: Bandwidth Requirement and State Consistency in Three Multiplayer Game Architectures. Proceedings of the 2nd Workshop on Network and System Support for Games, 2003, pp. 52–59, doi: 10.1145/963900.963905.

[21] Peterson, J.—Krug, J.—Zoltu, M.—Williams, A. K.—Alexander, S.: Augur: a Decentralized Oracle and Prediction Market Platform. 2015, pp. 1–16, arXiv: 1501.01042, doi: 10.13140/2.1.1431.4563.

[22] Rivest, R. L.—Shamir, A.—Wagner, D. A.: Time-Lock Puzzles and Timed-Release Crypto. Technical Report, Massachusetts Institute of Technology, 1996, pp. 1–9.

[23] Deering, S. R. H.: Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, RFC Editor, 1998.

[24] Shafagh, H.—Burkhalter, L.—Hithnawi, A.—Duquennoy, S.: Towards Blockchain-Based Auditable Storage and Sharing of IoT Data. Proceedings of the 2017 on Cloud Computing Security Workshop (CCSW '17), Dallas, Texas, USA, 2017, pp. 45–50, doi: 10.1145/3140649.3140656.

[25] Stoica, I.—Morris, R.—Karger, D.—Kaashoek, M. F.—Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. ACM SIGCOMM Computer Communication Review, Vol. 31, 2001, No. 4, pp. 149–160, doi: 10.1145/964723.383071.

[26] Superdata: Market Brief – 2018 Digital Games and Interactive Entertainment Industry Year in Review, 2019.

[27] Vorick, D.—Champine, L.: Sia: Simple Decentralized Storage. Nebulous, 2014, pp. 1–8.

[28] Webb, S.—Soh, S.—Lau, W.: RACS: A Referee Anti-Cheat Scheme for P2P Gaming. Proceedings of the 17th International Workshop on Network and Operating Systems Support for Digital Audio and Video, 2007, pp. 34–42, doi: 10.1145/1542245.1542251.

[29] Yahyavi, A.—Kemme, B.: Peer-to-Peer Architectures for Massively Multiplayer Online Games: A Survey. ACM Computing Surveys (CSUR), Vol. 46, 2013, No. 1, Art. No. 9, 51 pp., doi: 10.1145/2522968.2522977.

[30] Yan, J.—Randell, B.: A Systematic Classification of Cheating in Online Games. Proceedings of 4th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames '05), 2005, pp. 1–9, doi: 10.1145/1103599.1103606.

[31] Zyskind, G.—Nathan, O.—Pentland, A. S.: Decentralizing Privacy: Using Blockchain to Protect Personal Data. 2015 IEEE Security and Privacy Workshops, 2015, pp. 180–184, doi: 10.1109/SPW.2015.27.

Aleksandar Tošić is Teaching Assistant at the University of Primorska, and Research Assitant at InnoRenew CoE. His main research interests are distributed systems and distributed ledger technologies.

Jernej Vičič is Associate Professor at the University of Primorska, Primorska Institute for Natural Sciences and Technology in Koper, Slovenia. His main research interests are distributed systems and natural language processing.