Document hash proof of existence:

Hash: 5626ee12c748ae23ca110ce890465c734f7c9d145e090562ec232b4cfcd164a0

Transaction ID: 3d22a328268db9c2fa1083ba3a0bbc52085c08b13489725a8d7e25b8f2928e0f

Draft - Pokereum: An Efficient Smart Contract Dependent Decentralized Poker Platform

Oladapo Ajayi (ola)
oladapo@protonmail.ch
www.pokereum.io

Contributors:

Technical Advisors: , (TBD)

Version: 0.4.0

Abstract. A secure peer-to-peer (p2p) version of traditional online poker would allow trustless and provably fair poker games. This p2p approach would reduce the overall cost to players and add more value through incentive schemes designed to reward higher player participation. The Ethereum Smart contract system is a fully decentralized programmatic set of contracts using advanced blockchain technologies;[ref. 5] that will power one part of this solution; and Telehash [a secured network real time mesh technology; ref. 28] based, peer-to-peer poker network consensus provides the other major component.

We propose an alternative solution to the traditional mental poker problem by using a novel combination of smart contracts and meshed based player networks. These p2p networks will form consensus, which is secured through the Ethereum blockchain by the economic majority stake and irreproducible activity.

During a poker hand, Individual player actions resulting in state changes are broadcast to all opponents at a table during the game session and verified through a challenge response protocol by pools of statistical models of the poker network called jury pools. The pools then verify the table opponents' copies of the current player broadcast against their received copies.

Jurors (randomly selected players from the p2p poker network based on high stake and activity) are grouped into pools routinely, and replaced at regular time intervals by a system of contracts on the Ethereum network.

Each node selected to a pool explicitly uses its stake as collateral and risks losing its stake if it breaks the network rules. That incentivizes fair play and creates a disincentive or very imbalanced risk/reward offering for potential cheaters. As long as the p2p network is sufficiently protected against large numbers of player nodes under the control of a single person/group (known as Sybil resilience), the poker consensus protocol can safely agree on the states of the games, the state of all player accounts, and therefore the state of the network.

The Juror Pools will also agree on checkpoints to be permanently inserted in the parent blockchain each Epoch (a set of time-bound data allowing a summation of time that is divisible in 1 hour, 15 minute, and 1 minute increments). Dynamic members of the p2p poker network are updated using weak subjectivity [ref. 10] to the parent Ethereum blockchain. That means that nodes rejoining the network query and get the most recent epoch (last hour) of the network by querying the parent chain, then do the manual calculation to get up to speed on the current state of the network. That way nodes can be dynamic members of the p2p network leaving and rejoining the network with the guaranteed most up to date and secure state of the network, secured by the economic majority and activity of the network.

The combination of: decentralized and randomized selection of jurors (via Ethereum smart contracts), decentralized mesh networks for p2p game interactions, and heavy use of advanced anti-cheating mechanisms will, by this submission, present the poker-playing community a new and fun platform for playing poker without having to trust any individual/single entity. This paper is intended to describe the basics of a fully decentralized poker game experience.

1. Introduction

Most contemporary online poker games rely exclusively on trusted third party game hosts to facilitate an environment for secure real time gaming. That means that with any current online game, all players at a table are trusting that the hosts of the game (PokerStars.com, Titan Poker, etc.) are honest [reasonably believable] and capable of enforcing that honesty activity in all their employees [far less believable].

While that model is the status quo today, it suffers from the disadvantages of trust-based models. Game hosts claim that they can offer a trustworthy environment, but recent scandals have shown us that super admins are known to have taken advantage of unsuspecting players at poker tables; unfortunately this kind of malicious abuse of trust is both well documented

Примечание [1]: thanks

and reportedly widespread [ref. 18-19]. That has left many players skeptical as to the legitimacy of any trusted third party host claims [ref. 20-21]. In spite of the questionable operations of trusted third party online game hosts, many players still choose to participate in these games - often with other adverse side effects.

Such side effects may include: payment processing issues, high processing fees, high table rakes, being subject to antiquated regulations [ref. 31]and payment problems (see black Friday), geolocation issues, and poor software which comes with it's own set of trust issues/requirements. The last point is especially important due to the economic incentive for administrators within a centralized trusted third party host to benefit from an unfair knowledge of software algorithm functionality and game activity.

An even bigger problem is collusion. Game hosts spend a considerable amount of resources to reduce the effect of colluding players, but in all of today's centralized poker systems, collusion has not been eliminated completely. Moreover, in any trust-based model we cannot have a guarantee that a game host can completely eradicate collusion (in large part do to the onerous economic costs and their effect on operational margins). What poker players need is a trustless system using cryptographic proofs instead of trusting closed-door security policies managed and enforced by, yet again, trusted third parties.

The proposed trustless system will replace trusted third parties with autonomous agents whose functions are consistent and transparent to everyone. Autonomous agents (or smart contracts) can deliver cryptographic outputs, which can be viewed by any entity. Such a system is best served by using decentralized currencies (which have minimal transaction costs and no restrictions on the economic activities of the their users). Such a system will also dramatically reduce the amount of fees and rakes for participation, resulting in less direct costs to play for players and improving player potential in the multi-billion dollar per year online poker industry.

Traditionally, even the most expensive security mechanisms are vulnerable to the human component of collusion. By addressing collusion as a primary problem and using hardened Sybil resistance mechanisms (explained below) we have a basis for a secure and trustless p2p poker network. In this paper we propose a solution to the mental poker problem by using statistical modeling based consensus, collusion proofing, Sybil resilience, and autonomous agents or smart contracts. What we have is a fun and secure p2p poker consensus system that attracts players with very low costs, easy to use funding, and a logistically challenging and economically expensive system to cheat. What we have is: Pokereum.

Примечание [2]: Please see my comments below regarding collusion - I believe that it is easier to collude in a decentralized system.

Примечание [3]: + any more mentions of the words collusion or collude in this document.

Примечание [4]: In other words: while collusion should be addressed in this paper, I doubt we will reach a convincing argument why Pokereum is more collusion-resistant than centralized systems. Therefore, collusion resistance should not be touted as an advantage of Pokereum.

Примечание [5]: I suggest that you understand the whole paper by reading the entire document first. Then you can accurately come to an informed conclusion. Its rather premature to be making preconceived assessments based on prior paradigm of thinking. The entire documents has a combination of ideas that work together to provide reasonable solutions to several of the problems described in the paper

2. Parent chain

Although it would be possible to handle the security and long-term consensus within the poker network nodes exclusively, this is unnecessary and inefficient. Rather, we can leverage established and promising projects that allow us to offload some of the burden in developing the Pokereum network. That allows us to enjoy the benefit of a fast and secure robust blockchain while keeping the bloat [unnecessary consumption of resources] on the poker nodes low, essentially only needing a mini blockchain [ref. 29] (in this case a ringed state chain) that is weakly subjected to the parent chain [ref. 10]. There are two projects that fit the criterion as a trustless computing environment backend and resource for the Pokereum poker network.

The first is Ethereum, the second being Nxt. At this time Nxt plans to have smart contracts implemented sometime further in the future. Ethereum on the other hand plans to launch in the near future and already has a live test net available to builders for testing purposes. The reasonable strategy at this time is to start Pokereum on the Ethereum network while planning to accommodate Nxt integration in a future release.

The second set of projects fall within the area of decentralized storage options such as Storj, Swarm, or IPFS. The combination of a trustless computing environment and a distributed file storage system means we can build a poker system which will carry out most of the secure poker functionality and also provide the mechanisms to read and write data to long-term storage without any need to trust a single provider or host.

3. Anti Collusion

Collusion is the act of several players getting together to plan an attack on an unsuspecting player (or players) at the same table. The attack is simply a group of people collaborating to use a collective unfair advantage against an unsuspecting player leading to a situation where the cheaters alter the odds of play outcomes to their favor. To combat this problem we make sure all players can only play one game at a time by using random table placement. Every player at a poker table self-selects to that table through a random placement process between the ranges of several table options of the same table-stakes, grouped into the same category. That selection process happens on the front end. On the backend, an Ethereum based solution called Whisper [ref. 23, 25] does the job of organizing data called envelopes or messages into structures which our player network nodes can understand. These messages contain critical information related to each game session and the process of joining a game requires authentication to the poker network.The actual process of a player joining a certain table within a group activates a call to the current consensus mechanism of the network. The consensus mechanism of the network is a dynamically selected subset or statistical model of the number of active nodes in the poker network based on account balance and proof of activity/participation score acquired over time. The models serve the tables in the network by verifying network wide shared objects where needed through challenge response mechanisms.

Примечание [6]: depending on Ethereum/Nxt maturity, another strategy is using Bitcoin/Factom as fast, cheap and durable proof of existence, while keeping the state of the system only on Pokoereum nodes.

In essence, this system could resemble Master/Omni/Counterparty protocol. This assumes a convergence on the single most-popular blockchain is a significant advantage, I believe this option should be listed for completeness, even if it ends up not being selected due to technical considerations.

Примечание [7]: 1. We could have any number of strategies each adding its own extra layer of complexity, also extra barriers for our technical minds to navigate and implement, I am not opposed to including alternatives If there is a REAL need for them.

Another thing is the technology stack is already bulky and the bitcoin blockchain does not satisfy the needs of the real time requirements of the poker network and interoperability with a fast working test net and well documented smart contract system like Ethereum, even though bitcoin could be used ...

Примечание [8]: Each of these should have a references in the bibliography linking to the official project pages.

Примечание [9]: This is a very bad user experience. You cannot play poker on only one table. Poker players like to play between 4-16 tables at one time. depending on age and how fast their brains work. If you create a game that can only be played on one table then ...

Примечание [10]: This is just to start of course.We would tackle the multiple table issue as our system matures and when we cross that bridge

Примечание [11]: Right, however its a big deal. Again suggesting fast poker here (get moved to a new table every time you fold or the hand is over and start a new game as soon enugh players are available), as it solves many security problems, and increas ...

Примечание [12]: I agree, but we need to start small. A reputation system could help once we have started to bootstrap the network. I like your input here. whats your email?

Примечание [13]: This is very good. Random seating is fair and secure.

When a player attempts to join a table, they must complete a response to a challenge issued and validated by a selected subset of the player network (known as jury pools). That subset of network nodes check the network wide shared player object containing the player's history of current games to see if the player's node unique id is currently playing at a table. If found to be at a table the request is denied, if not the player is seated at a table chosen randomly within that table stake group. That's what happens on the front end.

On the backend players establish a mesh using links sharing a game session and participating in a open multiparty computation of shared game instance on each node. In other words, every node at a table knows the state and what actions of every other player node at the same table are eligible to take each turn, each turn verified by juror pools.

Once we can guarantee that one player node can only play at one table, we can then move to make it economically prohibitive and functionally a losing proposition to replicate player nodes across several computers.

Примечание [14]: losing?

Additional strategies to limit collusion could be employed by requiring player stake to be used as collateral during games and limiting the use of that stake to the current game, making it costly to spin up additional nodes on new computers.

Yet another strategy is to deploy low expected value dispersible automated software. Basically software that mimic human player actions, but are easily replaced when a real person joins a table. The last strategy will work very well at low capacity. We explore the characteristic components to this form of resilience in detail in the section below.

4. Trustworthiness and Sybil Resilience

The first part of the solution in limiting a player's ability to replicate nodes across the network is to use the player's account effective balance [the stake or monetary value they have invest in the network] in combination with the rate of spending (henceforth referred to as "account velocity") from that account. That calculated metric becomes a publicly accessible unit of trustworthiness across the player network. In essence, the higher the effective balance and velocity of a player node the more reliable the information relayed by that node is considered. That reliability also improves the likelihood that any particular player node will be selected to join a jury pool (elaborated on below).

The velocity of an account is defined as the frequency with which an account spends its balance within the poker network. The effective balance is the account balance after a prescribed waiting period. The effective balance does not affect game play or the ability to

spend stake; rather it is used to determine eligibility for jury pool selection. That's a variation of (and inspired by) the NXT "proof of stake model" [ref. 7].

In this scenario, an effective balance activation is approximately a quarter of an Epoch (see section 6 below) or 6 hours after a player node is funded. The purpose of the waiting period prior to the activation of the effective balance is to throttle the eligibility of new player nodes taking part in the consensus mechanism.

The account velocity or proof of stake velocity property [ref. 14] has two nice features that adds to the node's credibility while increasing difficulty for an attacker to replicate 'trusted' nodes across several computers:

1. The account balance velocity is used as one of the metrics to determine the level of trustworthiness of that node within the network.
2. That velocity or activity of the balance is also a measurement of the manual work done by the player on the node since most poker actions involve spending or betting some money from the balance.

For example: a call or a raise is an action initiated by the player by pushing the appropriate buttons dispensing some amount of energy for each action. These actions results in the depletion of a player's balance majority of the times, so most players will be cautious about the way they play. This energy or action is recorded as a variable in the account velocity function and is a part of the equation used to determine player rankings within the network. A higher velocity improves the probability that a player will rank higher within the network. It also improves the node selection as part of a dynamic statistical model.

Thus, by using a combination of proof of stake and account velocity we can have statistical confidence that the cost prohibitive and physically tasking requirements will be a great disincentive to any attacker in scaling beyond one poker node.

In a hypothetical network scenario of 40,000 poker nodes with a total combined player balances value of $100 million, to overcome this, an attacker would have to control about $40 million and 40% of the entire nodes, which equates to about 1,600 player nodes. Any entity with enough wealth and human workforce to overcome such security mechanism will also have the power to defeat any leading proof of work or stake consensus systems in existence.

Accounting for the exponential costs to cheaters(both in monetary value and system complexity), we leverage incentive and resource limitations to make it both infeasible and undesirable to attempt any form of Sybil attack on the Pokereum network. We dive deeper into the general simple mathematical models and consensus mechanisms below.

In an economic system, the velocity of money in circulation can be modeled by the formula

Примечание [15]: The velocity is primarily a measure of much a player is dominating the game and also playing style. if I understand velocity correctly it means how much a player is playing? Players that win will play more, losers will play less. Losing players are not less trustworthy I think it is more a function of how much of a "regular" or "pro" one is. I.e. a GTO bot would be able to play 24/7 and have high velocity? Or is velocity measured per hand? Could we define the term more clearly?

We do not want to encourage good players or bots, in fact we want encourage bad players that donate into the ecology if at all possible. We can also assume they are more trustworthy. In general we only need to worry about winners when it comes to any cheating. So instead of velocity we should measure how much players are losin ...

Примечание [16]: Velocity is simply the measure of frequent balance spending, it is fundamentally a neutral component. It has nothing to due with the expertise level of the player. Bad players or "fish" can still benefit from this quantification even if they lose all the time. See the gamification section on how this value helps player win perks from all the points acquired.

Примечание [17]: Velocity is what a poker player would call "action". You reward players who play more by allowing them get extra incentive. Generally high volume players are winning players therefore you are incenting winning players. These players however are bad for the game economy (because they take money out of it) you should incentivize the depositing players.

My worry are bots. Once you develop a winning bot, this bot would be able to get a high status in the network (ever ...

Примечание [18]: The algorithm to be developed at this point is general but it could be improved to account for deposits and frequent losers rewarding every loss with a higher number of points which these players can then exchange for various number of perks on the network. The point is at this time, this is a general specification and input like yours can be invaluable to encourage more participation by amateur players. But the specifics of these details need to be provided as set of options by board members with particular poker experience and then ...

$$V_T = \frac{TP}{M}$$

Where

V_T is the velocity of money in a given time frame
T transactions aggregate over a given time frame
P Is the price level
M amount of money in circulation [ref. 15]

We can apply a customized version to our player nodes accounts like so

Where
$V_{T'}$ is the velocity of player balance
T' is the total number of player node account transactions over a 1 min time period
P' is the average price of player spending at the tables
M' is the player account balance / stake

Therefore

$$V_{T'} = \frac{T'P'}{M'}$$

Where $V_{T'}$ is the velocity of an account and measure of observed activity

If we define the proof of stake velocity (V_P) as

$$V_P = \left(\frac{T'P'}{M'}\right) \times M'$$

We have a general model for the proof of stake velocity (V_p) that can be used for the ranking in the economic majority and physical activity

Another side effect of the general equation is that the components of T' are used as inputs to the achievement point calculation of each player node. We discuss this aspect in more detail below in the Gamification section. From now on we can associate V_p as a player's unit of ranking as determined by any arbitrary participating agent (aka "node") within the poker network.

5. Provably Fair Random Number and Shuffling

Mental poker is a set of problems that arise when dealing with playing card games over a distance without a trusted third party [ref. 3]. Invariably we are asking how can we guarantee a player isn't cheating when we all agree to follow a set of rules? Putting, aside the old server-client model [in which all trust is placed in the server host for security and fairness] there are three known approaches to the mental poker problem:

1) We have the shuffling algorithm with commutative encryption, to explain briefly this process involves passing card decks back and forth between players shuffling and using commutative encryption schemes.
2) A non-shuffling protocol in which players generate subjective random numbers for dealing card using the properties of homomorphic encryption to detect collisions. That's useful so that players only perform computations on cards that are only used in a game, while detecting already dealt cards.
3) Multi - trusted third parties: An improvement on the old server-client model. That involves making an implicit assumption that two or more third party servers used in the shuffling and choosing of cards are non-colluding [ref. 3].

All three of these methods have disadvantages that present bad user experiences or large economical risks to players.

Provably fair random numbers are generated by a system of Ethereum contracts with a combination of input data from several sources to improve the entropic seed.

1. The set of contracts in charge of this process receive a combination of all the seeds generated from a jury pool. The jury pools are computational models of the poker p2p network.
2. The seed combination is XORed with future Ethereum block hash in multiple steps, each with different sets of randomly incremental nonce values to generate a batch of random numbers for a shuffling step.
3. It then returns the storage location of batch random numbers to the next contract it calls - the "shuffling and encrypted card permutation" set of contracts.

4. That contract uses the return value of the random number batch contract to generate a sufficient number of encrypted decks. Each deck with a different permutation and each deck differentiated through the encryption and the unique one-time nonce per deck.
5. All decks are stored together with a commutative-shared secret of themselves.
6. The sets of shuffling and encrypted card permutation contracts then work with sets of storage and retrieval contracts to store and retrieve decks as needed by the network. They cache encrypted decks by the poker network in order to optimize operational costs within the Ethereum network.

Once we can guarantee the provably fair nature of each hand it becomes trivial to implement the retrieval and dealing of cypher decks and keys, which could be implemented in any number of ways using shared secrets and partially homomorphic encryptions. This would secure the community cards (for example) while determining whether the cypher decks or keys have been changed at any point in time.

6. Multi-Networks and Player Nodes

To understand the composition of each player node it's important to know how the protocols on which all player nodes are dependent and work together. Every node in the Pokereum network relies primarily on the Ethereum network, the Whisper protocol and the Pokereum Telehash real-time communication (RTC)driven network.

The Ethereum network is the foundation that each node is built on. Ethereum is a decentralized web3.0 platform that enables user generated smart contracts using Turing-complete languages [ref. 5]. Each player node communicates with contracts on the Ethereum network through an inbuilt Ethereum client in the Pokereum player node itself.

The whisper protocol is also an Ethereum based messaging protocol through which structural architectural information and specific game session details are routed for all player nodes. The whisper protocol allows for the organization of player nodes into groups or into an informational architectural framework in order to facilitate session based game play. The whisper protocol also acts as a transport layer helping high priority messages from consensus groups and nodes to propagate to all nodes on the Pokereum network.

The last major component is the Telehash driven component of the Pokereum network. Telehash is a protocol that enables strong encryption; mesh networking and RTC between endpoints [ref. 28]. In this case the endpoints are Pokereum player nodes. Additionally it is also planned that Ethereum-swarm or Ipfs decentralized cloud storage, when implemented will be utilized for long-term data store.

When a player downloads a Pokereum player node client, he creates an Ethereum external account within that client. Then in order to initialize and authorize the client for network

games, the player node sends a transaction representing its stake to Pokereum systems of contracts on the Ethereum network, which are in charge of initializing player node clients.

For each individual player node, once this stake transaction is received, another contract called the "player node agent" is created which is the contract equivalent of the external account on the Ethereum network. The contract adds several properties and functionality to the player node client and account that makes it autonomous with the rest of the contracts on the Ethereum network. Some properties added on to the player node and functionality or characteristics of the player node agent are:

1) The player node agent is accountable to the Pokereum system of contracts on the Ethereum network
2) The player node agent stores the player balance as network security stake; the balance is also used as needed by the player at each hand.
3) The Player node can request a withdrawal of its balance by sending a transaction in the form of a request to the player node agent. The request is then sent out and authorized by the Pokereum system of contracts after checking the state of the network, and validating the eligibility of the request.
4) The player node agent creates a number of player card encryption keys used in shared encryption of deck keys for table games.
5) The player node agent stores the reference location of the player card encryption keys on the distributed file storage network.
6) Player node agent is the reference by which the Pokereum system of contracts tracks and references each player node on the network.

Another unique functionality not specifically tied to the player node agent functionality is the ability of non-savvy users of crypto currencies to pre-fund accounts sending "Exchange requests" messages to other poker players on the network using the poker client messaging system. Such messages provide a template for users to structure the request and the methods in which they will complete the payment for the funds requested, alongside arbitration procedures if anything goes wrong. These messages are categorized as whisper "Fund request and exchange" topics [ref. 24-26] and available to any other poker players on the network actively listening for such messages.

Savvy poker players may capitalize on the user requests by adding additional fees for facilitating such exchange services. The exchange facilitators may accept offers by sending the required funds to a "Fund request and exchange" escrow contract [see system of contracts], which holds the user funds until the requester's payment is verified by the funder. Once payment is verified by funder, the new users account is instantiated with a player node agent. If something goes wrong arbitration procedure is enacted according to message template implicitly agreed on by the new user and funder.

Примечание [19]: alongside?

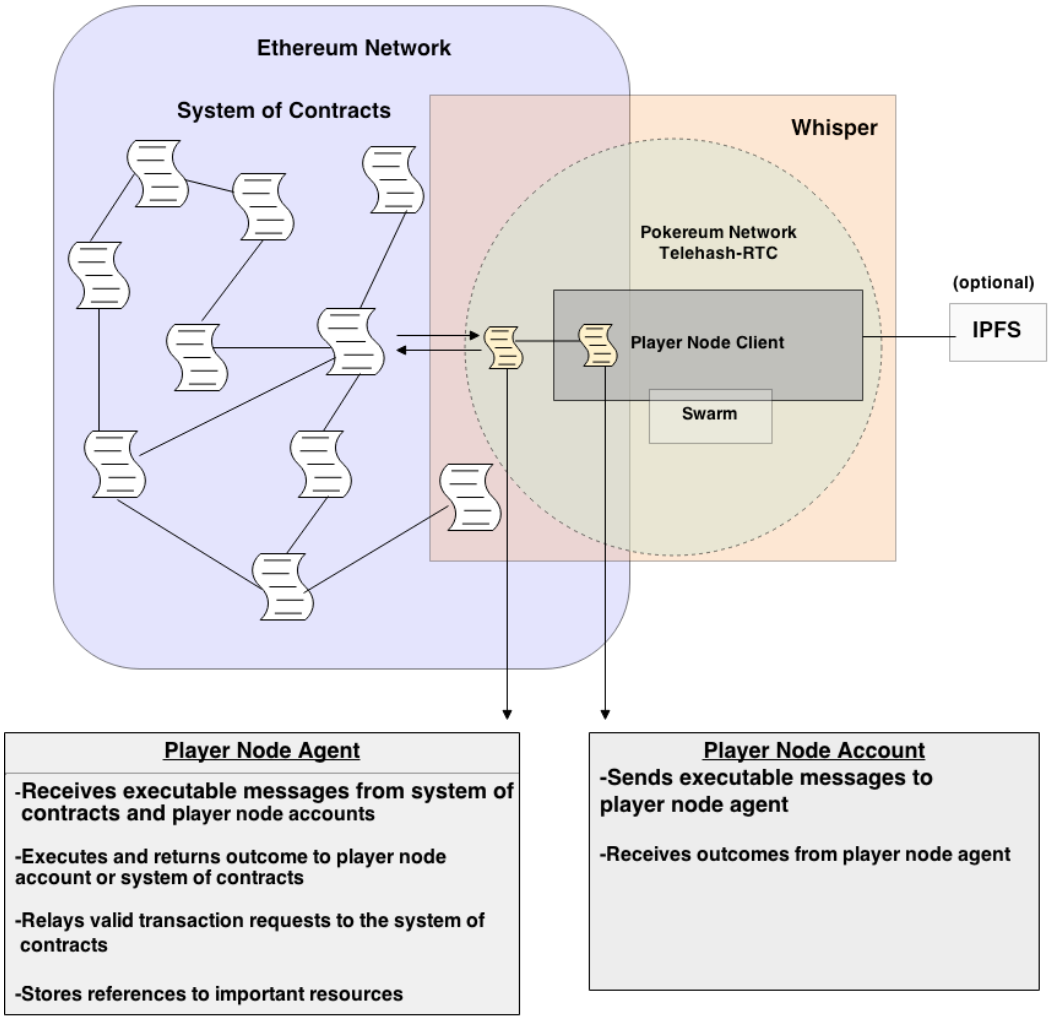
Примечание [20]: Just added- forgot to add an important detail about the Pokereum client. This exposes an interface to allow non-savvy would be crypto currency users to request funds for exchanges from other players. Basically it eliminates the on-boarding problem with acquiring crypto currencies. Also see system of contract section under "Fund request and exchange" for description of this functionality

A ranking mechanism of player nodes is important to the Pokereum consensus system no matter how dynamic the changes in ranking. As illustrated in section '3' above, the three heuristics combine to give each player node a dynamic economic and activity rank among all network nodes.

That ranking variable is referred to as the player velocity"V_P". The dynamic ranking V_P property is significant because its component combination is economically and physically expensive to attempt to replicate, hence it serves as a good measure of trustworthiness within the network.

Similarly the top two-thirds of the economic and activity majority nodes all share a high V_P and consensus among them constitutes a non-repudiable state of affairs of the network states.

Every player node in the network has private and public object properties. A player nodes' private object property includes a private public-key pair for 'Hashname' signatures shared by the Ethereum account creation contract. It also contains other private information privy only to that player node. Its public properties are objects and functions in a parent public object transparent to every other node in the network. For example the V_p, balance, message relay function, contract query function, contract transaction/message function, contract message relay function, "challengeBroadcast" function, "stateUpdateAll" function and many more. These objects are uniform across all player nodes and are invoked or used to achieve uniform consensus across the network.



7. Gameplay and Silent Jury Consensus

Jury pools are important to secure state transition of players account at each game or hand at every table. A jury pool is a group of player nodes with high " V_p" scores randomly selected each minute to relay critical network messages, verify each state change action of each poker node at a table or make blockchain based queries on behalf of the network.

A state transition is the change in state of a player's account parameters i.e. points, balance, account velocity etc. Each of the actions (i.e. bets or raises) that enable this transition must be acknowledged by table peers and approved by jury pools.

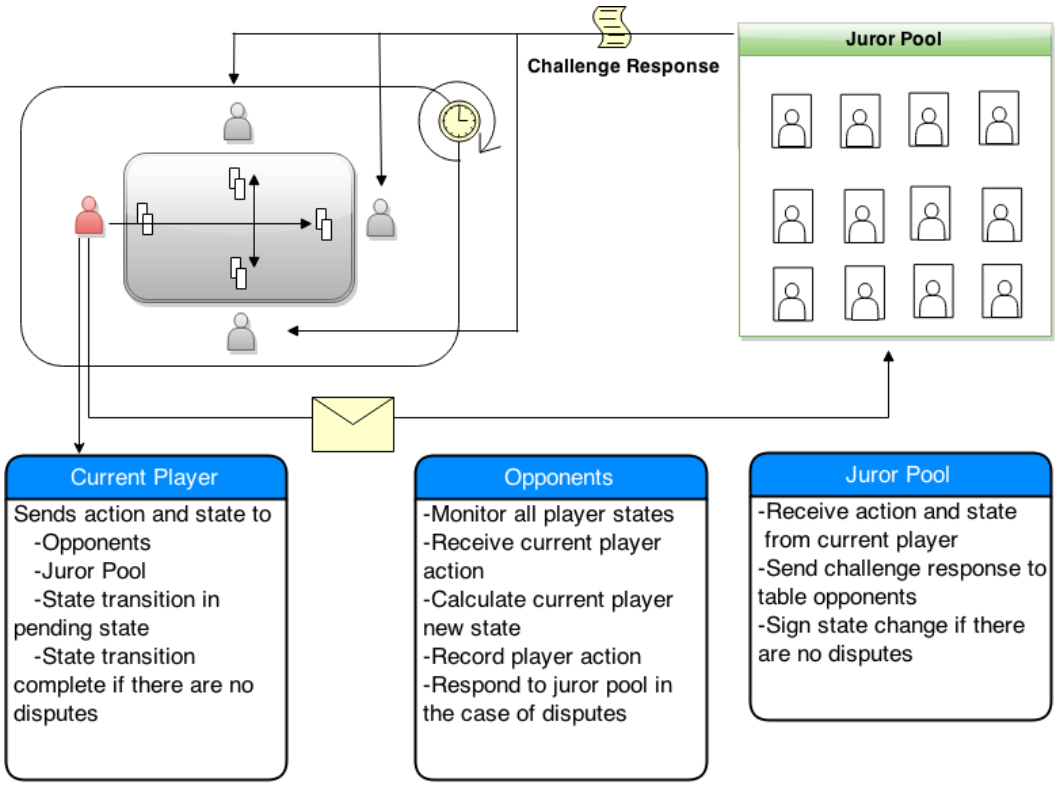
When a player node is placed at a table, the cards are provided as explained in 'section 4' and all nodes at the table play the same version of the game in the same session. Every player node simultaneously plays the same game using a simple version of an open secure multiparty computation with the related state transition enabled by sanctioned jury pools (see below). To be concise every version of the game on each node is the same game simulated on every other player node at the same table. For example before a current player "checks" every other player node sees and knows who the current player is and the action options of the current player. The current player node can only execute the list of action options available to it and every player at the table simultaneously manages the progression of turns to the next player.

During a player's turn, the action played is accompanied by player signature from the key pair verifying its first and last only move for that turn. That action is then broadcast as the legitimate action for that turn to the other table player nodes and the current Jury pools. The jury pools respond with a challenge response question as to the state of the last player broadcast and account. That challenge does not need any response from the other players if the information they received from the player is the same information from the jury challenge. Otherwise an alert is raised and the broadcasting player is punished.

The punishment may range from losing a round to getting kicked out from the table and losing the table stake depending on the frequency of offences. There is no way to cheat without breaking the rules of the network. Breaking the rules of the network even if unsuccessful means losing one's stake as explicitly stated by the network rules and understood by each player.

Since every player state change action (i.e. bet, raise etc) is signed by that player and publicly monitored by every other player node at the table and sanctioned by jury pools, it is highly impossible for a player node to misrepresent an account state. The nodes in the jury pool only last for 1-minute duration after which they are replaced by other nodes eligible to participate as part of the consensus mechanism of the network.

The name "Silent Jury Consensus" is appropriated because the challenge response validation checks by the jury pools runs in the background with no response required unless there is disagreement in consensus among player nodes at the same table.

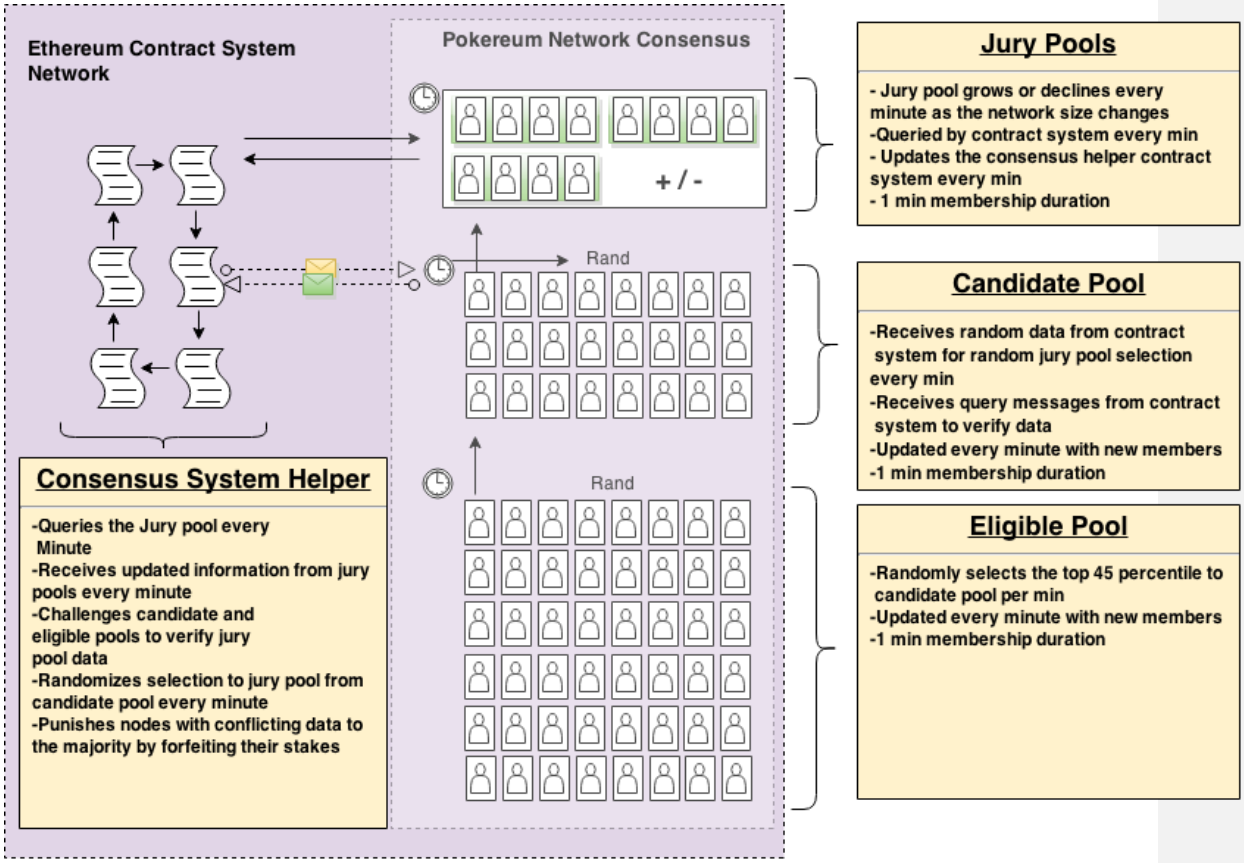


7.1. Jury Pools

Knowing the composition of each player node and how the V_p value works we can now explore the state change mechanism from the game play in relation to the consensus models. Each player node in the network has the potential to be part of the network's model of consensus called "jury pools."

The jury pools are groups of player nodes with high V_p scores. The number of jury pools grows and shrinks relative to the size of active nodes on the poker network. Each jury position only lasts for 1 minute as indicated by the parent blockchain time. After each minute the jury pool is disbanded and another set of eligible player nodes become active to form a new pool. There are three steps to becoming a jury member on the Pokereum network; each minute the composition of the pools is reset by a new set of nodes:

1. There is the initial eligible list who are a large group of nodes with high "V_p," eligible to contribute to the networks services such as relaying critical network messages, verifying each state change action of each poker node at a table, receiving blockchain messages or making blockchain based queries and transactions on behalf of the network. All nodes use the public object property to maintain this list.
2. The next step up is the candidate pool. This is a randomly chosen subset of the "eligible pool". The candidate pool is the pool where the next line of active jury pools is chosen, it's also a shared network wide object by all nodes.
3. The last pool is the jury pool. This is where the active governing of state changes within the network takes place. The pools are shared publicly across all nodes. Unlike the previous 2 pools, there are multiple jury pools. Every minute, the mechanism for electing nodes to the jury pool is initiated and completed by one of the set of external Ethereum contract calls to the network jury nodes. The call sends a random set of numbers based on the range of potential jury members in the candidate pool. That's how the next group of jury is selected from the candidate pool to the jury pools.



Once we have a mechanism to choose the most trustworthy of nodes we can now explore the mechanism by which the network state is secured. It is important to note that the nodes selected to jury pools are in charge of critical network messages, signing and updates, and also risk losing stake by signing invalid data or breaking the rules of the network.

Data can be deemed invalid or valid by using the state transition function applied to all time stamped and signed transactions and their corresponding accounts since the last checkpoint, determining the account and network states then comparing the root hash of the network state against that of the majority of other trustworthy nodes.

Player nodes in the candidate pool also validate every update, message or data signed by the jury pool. The general mechanism is to attain two thirds of the majority consensus in any of the pools. That is to say consensus on two-thirds amongst the nodes in the pool of the general order of time-stamped signed transaction by the each player node at each game session and the effect on each player node state.

If at any time the majority in a pool disagree on the state of the network the validation moves down the pool hierarchy down to all the nodes on the network. For example any disagreement in the jury pool triggers automatic revalidation by the candidate pool. If consensus still can't be reached, the eligible pool is automatically used and so on down to all the player nodes on the network. If and when the true state of the network is resolved and an incorrect signing of state change is discovered, the node initiating such an incorrect change and any other nodes supporting the incorrect change are punished by forfeiting their stake and rewarding the upstanding citizen player nodes in the pools or the network with the forfeited stakes.

7.2. Checkpoints

Assuming we have a secure network with reliable state of accounts, we must now decide how the state of the network is secured long-term. To limit the amount of data maintained by the Pokereum network, Long-term blocks of transactions and state trees are stored on a distributed file system and the last insertion is marked as a checkpoint reference on the Ethereum blockchain.

Dynamic members of the poker network use the checkpoints as a secure starting point to download blocks and get up to speed on the current state of the Pokereum network. Only the last block from the last 24 hours is stored on the Pokereum network. The 24 hour number is arbitrary but in reality only 2 hours worth of blocks are needed since every block references a new state of the network which is all that is needed to rectify all balances and other properties of all player node accounts.

Player node accounts are stored in network wide objects called a tree-forming account trees [ref. 29]. In this case the Merkle Patricia Tree since it is more efficient storage than ordinary radix trees. Also, we choose Merkle Patricia trees because it is developed and actively maintained by the Ethereum development team.

Account trees store and update the state of accounts in the network by incrementing or decrementing balances and other player node properties, using authorized time stamped node signatures, authenticated by the jury pools using challenge response mechanisms. The combined root hash of all accounts on the network serves as the current documented state of the network.

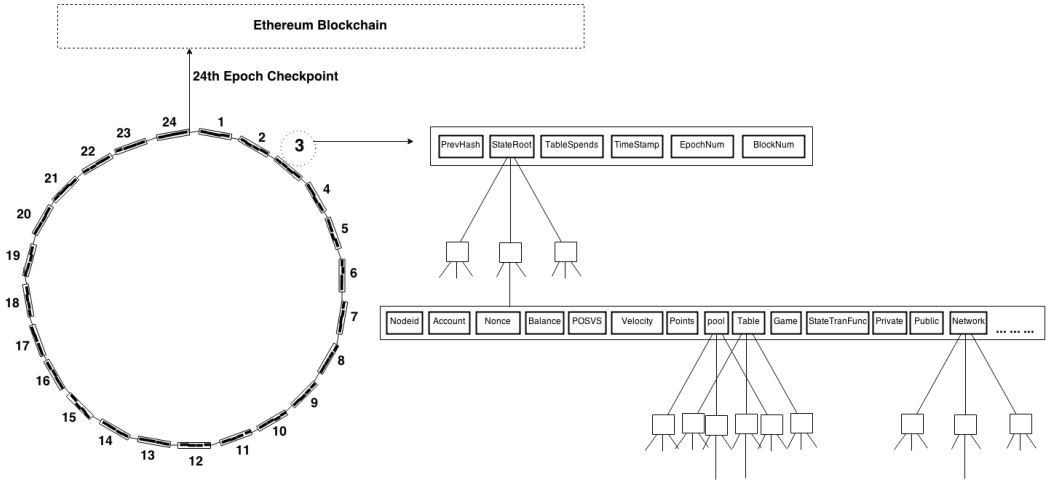
7.3. Epochs and Ringed Blockchain

Примечание [21]: I would recommend you read my and Vlad's scalability proposals from https://blog.ethereum.org/2014/11/13/scalability-part-3-metacoin-history-multichain/

Every hour a new block is solidified as the current state of the network and a checkpoint reference is inserted in the Ethereum blockchain while the actual block is stored into the distributed file store by a node in the jury pool and validated by its peers. The block inserter node is randomly chosen amongst the nodes jury pool. All jury pool members with valid signatures on the correct blocks are rewarded with a small fraction of the network table rakes each hour. The block to be inserted is compared and validated against all jury pool block versions and signed off on by at least two thirds of the jury pool members. Only then is the insertion operation allowed by the set of Ethereum contracts facilitating the insertion.

That happens every hour and this 1 hour duration is called an 'Epoch'. In this time based scheme there are Epochs, Sub-Epochs and minutes. Sub-Epochs are 15 minute timeframe checkpoints in the 1-hour long block in which the root hash at that moment represents the current state of the network. The 15-minute Sub-Epoch root hash is noted and signed by two thirds of the jury pool serving as a mini irreversible state of the Pokereum network.

Then there are also minutes. Minutes are 1-minute durations that determine the state of the network every minute by comparing root hashes. Every minute the new root hash comes about as a result of at least two thirds of the jury pool signing the root hash as the majority consensus. Because every 25th hour epoch is overwritten by a new 1st hour block on the Pokereum network we have an abstract blockchain ring, hence the ringed blockchain schema.



Each player node on the network is stored as a node in the Patricia tree and each node has several other attributes, which range from simple variables to complex objects and functions to other Patricia Trees for network wide data storages.

```
var playerNode = {
"nodeId" = "",
"hashName" = "",
"link" = "",
"path" = "",
"account" = [{},{}, {},…],
"nonce" = "",
"balance" = "",
"Vp" = "",
"velocity" = "",
"points" = "",
"pool" = [{ "jury": false},{ "candidate" : false},{"eligible":
false}],
"table" = [{},{}, {},…],
"game" = [{},{}, {},…],
"stateTranFunc" = [{},{}, {},…],
"network" = [{},{}, {},…],
….
…
..    }
```

8. Dropout tolerance

Dropout tolerance is the ability for a game to continue when a player decides to quit a game either politely or impolitely [ref. 1]. Since perfect abrupt or polite dropout tolerance can only be achieved with trusted third parties, replacing a trusted third party with automated trustless contracts is most viable method to achieve similar outcome. Essentially since smart contracts replace the role of trusted third parties, dropout tolerance becomes less of an issue due to the new architectural dependence on smart contracts.What that means is that players are not dependent on each player to reveal critical game information. In the old mental poker schemes, secret-sharing schemes were employed to reveal opponents cards when the game necessitated. This created the problem of dropout tolerance. We basically extinguish the problem of handling drop out tolerance in the mental poker paradigm by employing the use of

trustless smart contracts that are capable of revealing encrypted player card keys when certain conditions are satisfied.

9. System of contracts

The Ethereum platform enables the creation of autonomous agents or smart contracts. The contracts are executed simultaneously across the network nodes in the Ethereum virtual machine, when called by sending them transactions.

A contract is also a type of account, there are two types of accounts: externally controlled accounts and contracts. Externally owned accounts are accounts owned by external entities, which can be used to send transactions to other contracts to cause them to execute their code. Contracts or agents on the other hand are not controlled by anyone. These contracts are trustless, self-executing agents meaning, that they run their code when they are called. These trustless agents can also call other agents and receive a return value from the completed execution of other agents. There are four general categories of contracts that enable us to use them as an efficient system of trustless backend contracts for the poker network:

Примечание [22]: What is the other type of account?

Примечание [23]: two types : 1) externally owned account, which is simply your wallet, 2) and a contract which is a self executing account/contract with a balance and code which runs when a transaction is sent to it.

1) Contracts can serve as a software library providing functionality to other contracts i.e. encryption or storage provisioning.
2) Contracts can be used to maintain a data store. i.e. A list of accounts balances with membership access tokens.
3) Contracts can manage ongoing complex relationships i.e. multisignature accounts or escrow.
4) Contracts can serve as forwarding contracts; these are “externally owned accounts with complicated access policies”. That functionality can in essence require certain conditions to be met before forwarding a message to another destination contract [ref. 6].

The limit to the execution of a contract is a special unit called gas for every computational step. Every transaction must be sent with a minimum gas price and the total price to be paid for the total number of computational steps required [ref. 5].

The system of set contracts should be sufficiently encapsulated and modular to allow for destruction of depreciated contracts or use by other gaming networks. With all the category functions of the contracts and by taking gas cost into consideration we can build an efficient system of smart contracts that will work together to fulfill the security and storage needs of the poker network. We can now build a decentralized backend for the poker network. Some samples of the useful contracts are explained below

- **Library of contracts**: These are sets of contracts acting as the main forward access contracts. They maintain lists of all the system of contracts and forward all messages to the "Forwarding access" contract. The main purpose of this contract set is to maintain modularity and enable the depreciation of old contracts and inclusion of new ones. It also acts as an interface for other projects willing to use some components of the system of contracts.
- **Forwarding access**: These set of contracts fall into category "4". They will serve as a second step gatekeepers checking to see if an inbound transaction has the right parameters to run their code. That is, sending out and routing the transaction message to the right destination contract.
- **Challenge response**: These are a set of destination contracts that validate requests made by nodes of the poker network.
    - **Challenge response state checker**: These are sets of destination contract called by a "state transition" contract set. They validate the current state of the network that they receive against eligible randomly selected nodes in the list of accounts contracts. An account is eligible if it has a certain percentage of jury eligibility trust points calculated from the multi heuristic network data of a player object.
    - **State Transition**: These contracts set call the "Challenge response state checker" receives a return message and approves and cements the current state in storage as the most valid up to date state of the Pokereum network.
- **State provision**: This set will provide the last updated checkpoint of the network to any of the current jury nodes that request it. The checkpoint is then passed on to any new poker node joining the poker network.
- **Commitment random seed generation**: These sets of contracts collect a combination of all the seeds generated from all current jury peers, the total of which is XORed with future block-hashes in multiple steps each with a different round of random incremental nonces to generate a batch of random numbers for the shuffling step. It then returns the storage location of batch random numbers to the next contract it calls. [The: "shuffling and encrypted card permutation" contract]
- **Shuffling and encrypted card permutation**: This contract set is called by the "random seed generation contract" above. It is passed a return value, which is a reference to the storage batch of random numbers location. It gets the actual numbers and then proceeds to generate a sufficient number of decks. Each deck with a different permutation and each deck differentiated through the encryption and the unique one time nonce per each deck. All decks are stored together with a commutative-shared secret of the decks.

- **Card permutation storage**: This contract is simply a reference library to external storages of deck permutations and secrets. It returns information to the 'Card permutation provisioning" contract.

- **Card permutation provisioning**: This contract is invoked by a message from the poker network jury. The contract then calls the "Card permutation storage" routing required deck information from the storage space (ie swarm, Storj, Ipfs) to nodes in the poker network.

- **Exchange**: These are sets of contracts in charge of making exchanges between user preferred currencies to "stable value currency" and then finally to stable native Pokereum network token (cash) or vice versa. It works with the "alt payment integration" contract to detect what currency the user decides to play with and forward the request to the exchange. That way BTC, NXT and other currencies can be used seamlessly on the Pokereum network.

  The contract set is actually a series of network calls to the real exchange system of contracts called Etherex [ref. 13]. That way players can play with a stable monetary system while the funds are used on the poker network. Note that "Pokereum network cash" is actually just a representation of funds held and manipulated within the trustless contracts. It represents a credit on the system; actual user currencies aren't used on the poker network.

- **Adaptive**: These contracts enable different alternative coin community integration by extending an adaptive interface to allow any coin developer to write contracts which interact with the contract payment modules
  - **Stable value currencies**: These are contract sets that hold the final currencies used by the poker network. The Derivatives and currencies held in these contracts are stable in their value relative to the USD [ref. 5] and manipulated according to the Pokereum network state. Currencies like BITUSD, Nubit or even Ether and Nxt if stable enough can be the final currencies in which funds are held and manipulated in trustless contracts while reflecting the state changes on the Pokereum network.
  - **Alt payments integration**: These are a set of contracts that do as the name suggests. These are hundreds of contracts written by coin developers to enable any of their coin holders to play on the poker network. Basically each contract routes its payment to the exchange contract which turns the payment into a stable value currency which is then exchanged for Pokereum network cash/token.

- **Fund request and exchange**: These are sets of escrow specific contracts that hold and allow non-savvy would be users of crypto currencies to receive pre-funds from other savvy crypto currency poker players on the network pending payment verification confirmation. For example, funding requests initiated by non-savvy crypto currency users using the client messaging would include templates detailing the steps to pay for such fund requests.

The funds provider sends a release confirmation message to the “Fund request and exchange” contract once payment is made according to the funding request agreement. Otherwise arbitration procedure implicitly included in the user-funding request are activated. If all goes well and the funder confirms payment a transaction is sent to the “Account creation and authentication” contract set to initialize the new user account with a player node agent.

Примечание [24]: Updated this with important usability information for regular poker players

- **Account creation and authentication**: sets of contracts in charge of initializing a poker-nodes account and balances and acknowledging it as a proper member of the poker network by granting such node a secret. These contract sets create a “player node agent” contract for each account on the network. These sets of contracts can initialize a “player node agent” by receiving a confirmation message call from the “fund request and exchange” contract or by a transaction from an externally controlled account.

- **Player node Agent:** This is one of the most critical contract pieces that add autonomous functionality to each player node. Some of those functionalities are described below:
    - The player node agent is controllable by the Pokereum system of contracts on the Ethereum network
    - The player node agent stores the player balance as network security stake; the balance is also used as needed by the player at each hand.
    - The Player node can request a withdrawal of its balance by sending a transaction in form of a request to the player node agent. The request is then sent out and authorized by the Pokereum system of contracts after checking the state of the network, and validating the eligibility of the request.
    - The player node agent creates a number of player card encryption keys used in shared encryption of deck keys for table games.
    - The player node agent stores the reference location of the player card encryption keys on the distributed file storage network.
    - Player node agent is the reference by which the Pokereum system of contracts tracks and references each player node on the network.

- **Account tracking**: These contracts track the difference between the last state and current state and submit the result to the “Account/settlement” contracts.

- **Settlements / Accounting**: These contracts are called by the “account tracking” contract and passed the result of the difference of accounts balances between past and current states. The contract uses this difference to reflect the most up to data state of all network accounts balances. Reflects the proper state of network balances.

- **Account payouts**: These contracts are called by sending a message by any of the Pokereum network nodes. The message is verified and checked against the "Settlement/ Accounting" contract to if the message is eligible for its purpose the message request is processed, otherwise an invalid error message is sent to the poker node.

- **DAPP reserves**: These are essentially the Pokereum network net proceeds categorized into weekly groups. When they receive messages with the right properties and required access token information from externally controlled accounts, they process the request on the weekly group categories reserves. If the token security check is passed, a portion of the DAPP reserves of all the reserve weekly groups, proportional to the token amount in the sender account is returned to the sender of the message. If such a message has already been made or contains invalid information, the request is denied and an invalid error message is returned.

10. Player Retention and Gamification

Примечание [25]: Where is the economic model? Rake, fees etc.

Примечание [26]: This is not specified as of yet, as this is just an implementation detail to be decided by community consensus (DAO) as a whole. But in general it should be easy to outline, the rake percentages, rake-backs, tournament and private table fees, etc...for the DAO as a whole to decide on before implementation by DEVS

Because of the low cost of operations in comparison to other centralized poker hosts, some of the generated reserves could be used to reward the most loyal poker nodes. Each individual node is assigned a number based on a combination of some multi heuristic elements. This is the basis of a point system shared by all nodes and validated by the consensus mechanism. Each node would then be rewarded with perks as they ascend through the levels indicative by badges of 10 ranks. Each player node ranking level is not permanent and will need to be maintained over time, in essence its rank within the point system is maintained by an exponential decay function with a half life determined by the player nodes account velocity. The higher the account velocity, the higher the half-life of the player account. The actual player node point number is a number that grows based on player nodes action in several contexts. A different set of perks is then awarded to nodes at each level. The top ranking nodes are rewarded with table rake backs, free private tables, free tournament entries and much more as the network matures.

General exponential decay function

$$\square_{\square}{}'(t) = \square_{\square}{}'_{0}\,\square^{-\square\square}$$

Where

$P_{p}'(t)$ is halving of player points after time 't'

P_{p}' Is the initial point
P_{0}

General Player node point system

Using the components of the P_{p} variable from 'section 2 trustworthiness and sybil…'
We can assign points to each P_{p}component or account action resulting in a spend
And assign points to each game won, game wins have a higher point allocation

X is the number of player node transaction per minute
W is the wins per min
A[n] is an array index that holds player each player point per min

$$\sum_{n=0}^{x} A[n] + 100 + 250W$$

11. Long Term Viability and Management

Due to the demanding nature of online poker it is inconceivable that a viable open source decentralized poker project managed by a few individuals could address all externalities or other issues that arise in any poker operation. Because the Pokereum decentralized poker project is essentially a decentralized autonomous application or DAPP [ref. 30], a similar organizational structure is needed to manage all externalities, policies, and all sorts of operations including customer support, maintenance and development.

Luckily we live in an era where virtual online organizational structures of no physical or extra jurisdictional limitations are starting to materialize. An example of such a structure is a DAO or Decentralized Autonomous Organization [ref. 30], with its own delegate election procedures and operational policies formed to ensure long-term success and protect organizational resources.

Hypothetically, an entity becomes a member of the DAO by owning specialized access tokens to some of the DAPP's contracts or other DAO functionality. Members are also free to revoke membership by transferring token to other entities. Members may consist of individuals,

establishments and or autonomous agents, which enforce DAO member consensus or delegate actions. Every member of such an organization eponymous or anonymous is incentivized to contribute to the long-term success of the underlying DAPP.

Access tokens may grant access to reserves generated by the DAPP, which may be stored in specialized contracts, with each token mapping to its approximate portion of the contract reserves. Such an organization could then use a mild and different form of futarchy for its decision making process [ref. 12].

The difference from traditional futarchy is that, instead of members betting on the best possible outcome, they instead vote on the best decision option based on the useful objective information provided by delegates with related expertise. The delegates then enforce the majority outcome of votes. Essentially what we have is a self governing DAO that manages the operations of a collective, with an efficient decentralized decision-making process drawing intelligence from the masses to enable the long-term success of the underlying DAPP.

12. Conclusion

We have proposed a system to enable provably fair, peer-to-peer poker using autonomous agent or contracts instead of relying on trusted third parties or using the usual complicated and inefficient mental poker schemes.

We started by addressing the problem of collusion and sybil attacks using random table placement and jury approval using challenge response protocols. We also employ a multi-heuristic approach to make it economically expensive and physically challenging for an entity to scale beyond one node, deterring the potential issue of sybil attacks. That makes the Pokereum network robust and secure as long a sybil attacks are disincentivized and the conditions for participation make sybil attacks practically impossible.

Agents or trustless contracts shuffle and encrypt card decks in mass, seeded by a XORing of random seeds generated by nodes on the network. Encrypted cards are downloaded by all nodes and used as needed, eliminating used decks on each table use.

Poker nodes in a game participate in an open multiparty computation of the state transition of the game state verified by statistical model of the network acting as jury in a challenge response protocol to validate every update. The dynamic members of the network can freely join and leave the network getting checkpoints of the most current state from the parent block chain on their return and can be confident that their view of the network is the right one.

Furthermore, nodes use deposits or stake as one of the heuristic metrics for trust and could possibly lose their stake if they act against the network rules. Nodes with low probability

Примечание [27]: I am super impressed with this document. I hope it actually works technically. Online poker needs some help. It feels like there has been lots of technical coverage but I am not sure how the economic side of it works just yet.

One issue I see is that of bots, how do we assure one account per human and that only humans can play?

Примечание [28]: I am glad you made it this far to have a full picture of the solution. As with anything technical we will start by conducting experiments on a test net. But first on a simulation test net.

Using stake as bonds, the bot owners would have to be super wealthy, How many such entities have such funds to throw around? ;) an even if the odds of actually sitting 2 bots at a table grow exponentially as the number of players on the network increases

Примечание [29]: I assume the technical coherence of your proposed architecture - very interesting document btw - without having had the time to fully digest it yet. But a question: does the usability of Pokereum rely on the arrival of sub second blocktimes on the ethereum pubilc network? It seems that players may have to wait for several blocks for decks to be available for game play.

Примечание [30]: _Помечено как решенное_

Примечание [31]: _Открыто повторно_
Thanks for your question, the actual games respresent mesh based user sessions using RTC (Real time communication) by way of Telehash....Games are real time ie not dependent on ethereum blocktimes at all...So then the question becomes how do you handle security? Security is based on validation of a dynamic set of nodes on the network called "jury". These nodes use a combination of stake bonding, historical proof of activity, money spending velocity aided with the ether blockchain to be able to position themselves as validators in challenge response schemes similar to client server communication. results from each round of games are stored from time to time (Epochs) in the Ethereum chain, ie weak subjectivity ...

Примечание [32]: From the Ethpoker team eh? ;)

scores on each of the heuristics are less trusted than nodes with higher scores. This way only the most active provably human nodes with stake to lose are incentivized to be honest and will uphold the integrity of the network.

References

[1] Adi Shamir, Ron Rivest and Len Adleman, "Mental poker," http://people.csail.mit.edu/rivest/ShamirRivestAdleman-MentalPoker.pdf 1981.

[2] Tzer-Jen Wei and Lih-Chung Wang, "A fast mental Poker protocol," http://eprint.iacr.org/2009/439.pdf 2010.

[3] Sergio Demian Lerner, "Mental Poker Framework," http://www.dc.uba.ar/inv/tesis/licenciatura/2010/lerner 2010.

[4] Wikipedia, "Mental Poker," http://en.wikipedia.org/wiki/Mental_poker 2013.

[5] Vitalik Buterin, "Ethereum White Paper," https://github.com/ethereum/wiki/wiki/White-Paper 2014.

[6] Vitalik Buterin, "Ethereum-Development-Tutorial," https://github.com/ethereum/wiki/wiki/Ethereum-Development-Tutorial 2014.

[7] Nxt Wiki, "Nxt white paper," https://wiki.nxtcrypto.org/wiki/Whitepaper:Nxt 2014.

[8] Vitalik Buterin, "Scalability-part-2-hypercubes," https://blog.ethereum.org/2014/10/21/scalability-part-2-hypercubes/ 2014.

[9] Vitalik Buterin, "Scalability-part-3-metacoin-history-multichain," https://blog.ethereum.org/2014/11/13/scalability-part-3-metacoin-history-multichain/ 2014.

[10] Vitalik Buterin, "Proof-stake-learned-love-weak-subjectivity," https://blog.ethereum.org/2014/11/25/proof-stake-learned-love-weak-subjectivity/ 2014.

[11] Vitalik Buterin, "Schelling Dollar," https://github.com/ethereum/serpent/blob/poc7/examples/schellingcoin/schellingdollar.se 2014.

[12] Vitalik Buterin, "An introduction to Futarchy," https://blog.ethereum.org/2014/08/21/introduction-futarchy/ 2014.

[13] Etherex-crypto, "A decentralized exchange built on Ethereum," https://github.com/etherex/docs/blob/master/paper.md 2014.

[14] Larry Ren, "Proof of Stake Velocity: Building the Social Currency of the Digital Age," https://www.reddcoin.com/papers/PoSV.pdf 2014.

[15] Wikipedia, "Velocity of money," http://en.wikipedia.org/wiki/Velocity_of_money 2015.

[16] Wikipedia, "Nash Equilibrium," http://en.wikipedia.org/wiki/Nash_equilibrium 2015.

[17] HackApp, "Devp2p Wire Protocol," https://github.com/ethereum/wiki/wiki/%C3%90%CE%9EVp2p-Wire-Protocol 2014.

[18] Scott Clark, "The Largest Poker Heist In History," www.insidestl.com/insideSTLcom/STLSports/STLPoker/tabid/176/articleType/Articl eView/articleId/4503/The-Largest-Poker-Heist-In-History.aspx 2010.

[19] Chad Holloway, "Travis Maker Leaks 'god mode' Email from UltimateBet Scandal," http://www.pokernews.com/news/2011/03/travis-makar-leaks-god-mode-email-from-ultimatebet-scandal-9946.htm 2011.

[20] Steven D Levitt, "The Absolute Poker Cheating Scandal Blown Wide Open," http://freakonomics.com/2007/10/17/the-absolute-poker-cheating-scandal-blown-wide-open/comment-page-8/ 2007.

[21] Bruce Schneier, "Cheating in Online Poker," https://www.schneier.com/blog/archives/2007/10/cheating_in_onl.html 2007.

[22] Vitalik Buterin, "Merkle Patricia Trees," https://github.com/ethereum/wiki/wiki/Patricia-Tree 2014

[23] Juan Batiz-Benet, "The Permanet Web," https://github.com/jbenet/ipfs/ 2014

[24] Lokiverloren, "Whisper," https://github.com/ethereum/wiki/wiki/Whisper 2014

[25] Daniel A. Nagy, "Whisper Overview," https://github.com/ethereum/wiki/wiki/Whisper-Overview 2014

[26] Gav Wood, “Swarm,” https://github.com/ethereum/wiki/wiki/Whisper-PoC-1-Protocol-Spec 2014

[27] Naoki Tsujio “Webrtc- Chord, a Distributed Hash Table, Using WebRTC,” https://github.com/tsujio/webrtc-chord 2014

[28] Jeremie Miller, “Telehash,” https://github.com/telehash/telehash-js 2009

[29] J.D. Bruce, “The mini-Blockchain Scheme,” http://cryptonite.info/files/mbc-scheme-rev2.pdf 2014

[30] Vitalik Buterin, “DASs, DACs, DA and More: An incomplete Terminology Guide,“ https://blog.ethereum.org/2014/05/06/daos-dacs-das-and-more-an-incomplete-terminology-guide/ 2014

[31] Stan Higgins, “Bitcoin Poker Site Seals With Clubs Closes After Security Compromise” http://www.coindesk.com/bitcoin-poker-site-seals-with-clubs-closes-after-security-compromise/ 2015