A Holistic Blockchain Architecture for IoT Systems: Design Considerations and Challenges

Abstract—**This paper introduces an IoT Blockchain platform that encompasses a new decentralized consensus protocol named *Proof of Real-time Transfer* (PoRT) that addresses the challenges of privacy, security, scalability and storage in IoT systems. PoRT protocol preserves privacy by conferring decentralized identities to every entity in its ecosystem, encrypts data at source and preserves data integrity and authenticity using digital signatures. PoRT adopts a novel construct for blockchain storage that places no hard requirements on universal state, rather imparts tamper resistance through cryptographic sortition driven sub-chain replication across the network. Nodes maintains three local chains namely, a main chain, a custody chain and a log chain. While the main chain comprises blocks validated by the node and the log chain saves summary of services rendered, the custody chain comprises random segments of the main chain and the log chain from across the network. The above construct is highly scalable, is leaderless, and yields instant finality and tamper resistance to chains. PoRT combines the powers of cryptographic sortition and Kademlia DHTs in running behavioral audits placing the onus on the platform delivering on its data storage promises. Finally, this paper discusses newer challenges to tackle with formalized solutions in the space**

Index Terms—**IoT, blockchain, consensus protocol, decentralized storage, cryptography**

I. Introduction

Trust and transparency form the two core tenets of the Web3 vision, one that encompasses building secure, privacy-centric, equitable digital environments that represent a fundamental shift in how the world interacts today. The vision promises a democratized internet that distributes power and payouts among its many facilitators, while enabling authenticated communications and yielding transparency and auditability in services rendered. At the core of Web3 lies Blockchain technology, one that is built on the foundations of peer-to-peer networking protocols, cryptography, distributed systems and data structures.

As a network of interconnected devices and sensors, the Internet-of-Things range from smart home sensors to devices that cater to industrial applications and healthcare, collect, analyze and transmit data to end users. They are ingrained in every facet of our lives and invariably carry sensitive information that should not be censored, or breached, or lost in transition. Transitioning IoT services from centralized platforms to decentralized equitable architectures where solutions begin with preserving privacy, data integrity and data security and restoring data ownership at the hands of end users, can unlock the many untapped potentials of this sector. Such a transition, while naturally beneficial to end users, will invariably expand the solution horizon with newer value added services that benefit the solution providers as well. The opportunities are discussed below.

- Improved security: Web3 technologies face reduced risks of single point of failure, DDoS attacks and data breaches, due to decentralization and the adoption of cryptography.
- Interoperability: With standardized data schema, Web3 enables seamless communication and integration between different devices and IoT platforms.
- Digital ownership rights: Developing an NFT ecosystem that issues digital certificates of ownership and supports their monetization, creates newer streams of revenue.
- Smart contracts: Automated execution of complex business logic between two parties through smart contracts, reduces operational costs and improves efficiency.
- Decentralized data marketplaces: Users can sell and buy data at decentralized markets, through tokens, and support research, analytics and other business use cases.
- Token economics: Through the use of tokens, micro payments can be issued to incentivize collaboration and data sharing among devices or users.
- Community-driven development: Companies can leverage off of open-source tools, decentralized applications that are largely developed by developers world wide.

A. The IoT Blockchain design criteria

Supporting full-fledged IoT ecosystems on decentralized platforms entails designing a Web3 protocol that can handle high volumes of IoT data without compromising on real-time data processing, low cost, data security, data integrity and decentralization. IoT devices come in innumerable form factors serving niche applications, while facing notable constraints in one or more of power, memory, compute and network connectivity. The identified protocol should enhance device security, should secure data at source and be compatible with standard IoT protocols such as MQTT and CoAP. Importantly, it should be designed for energy efficiency and optimized usage of memory, compute and network bandwidth.

Apart from designing the right consensus protocol, identifying a well laid-out Blockchain architecture that is provisioned to support value added services in phases is key. The platform should be developer-friendly with support in the form of developer tools, APIs and SDKs to enable developers to build decentralized applications for IoT. Further, developer tools should encompass writing and deploying smart contracts and being able to periodically verify the integrity behind their execution. The architecture should spell out provisions for decentralized storage and its proposed support for analytics.

TABLE I
A SUMMARY OF POPULARLY RECEIVED IoT BLOCKCHAINS

Summary of Blockchain	**Consensus Mechanism**	**IoT Relevance**
IOTA [6]: Focuses on enabling trust-less machine to machine transactions. Adopts Directed Ayclic Graph like data structure called Tangle. For every new transaction added to Tangle, two older transactions are to be. validated. This forms a web of interconnected approvals gathering consensus on transactions.	Protocol is leaderless and has no miner or validators. Proof of Work (PoW) is employed in a lightweight manner. A *coordinator* node that the network *trusts*, plays a critical role in the prevention of double-spending.	Fast, feeless and highly scalable. Allows for simultaneous processing of transactions. Employs pruning to remove older transactions thereby managing storage requirements. Paved way for IoT device interoperability. Enabled tamper resistance through cryptography.
IOTA 2.0 [7]: Adopts Fast Probabilistic Consensus doing away with coordinator nodes. Offers improved resistance to a variety of malicious attacks.	Proof of Work requirements are adjusted dynamically based on network state and current demand.	Introduced support for smart contracts an upgraded wallet architecture for the supply of IOTA tokens.
IOT Chain [8]: Enables value transfer in IoT systems Adopts DAG structure that allows parallel processing and increased throughput. Encrypts data at source.	Uses hybrid consensus in Proof of Work (to select nodes) and Practical Byzantine Fault Tolerance (for fast finality).	Prevents data breaches, device tampering and cyber attacks with Trusted Execution Module (TEM). Supports smart contracts and Dapps.
IoTeX [9]: A DAO governed blockchain that aims to create Internet of Trusted Things (IoTT). They adopt Trusted Execution Environment (TEE) and Hardware Security Modules (HSM) with strong focus on privacy.	Uses a hybrid consensus in Delegated Proof of Stake (DPoS) and Roll-DPoS (introduces randomness), both known for being highly efficient and scalable.	Root chain provides the decentralized and secure infrastructure, while sidechains allow for high scalability. Supports smart contracts and building niche IoT applications.
VeChain [10]: The VeChainThor blockchain is designed to provide a secure, efficient and trustless product tracking and improved supply chain management. It is compatible and interoperable with Ethereum.	Uses Proof of Authority (PoA) consensus. New blocks are created by Authority Masternodes that are entrusted with the task based on their reputation.	Users IoT devices to track products from production to distribution to consumption Supports the creation of smart contracts, sidechains and custom tokens.
Hyperledger [11]: It is a permissioned blockchain that is highly customizable with a plug and play modular architecture. Designed for enterprise adoption .	Supports pluggable consensus, allowing for users to choose from different consensus algorithms based on need.	Used in supply chain management for real-time visibility and transparency. Supports smart contracts (chaincode).
Helium [12]: Is a decentralized wireless network for IoT devices. It uses Low Range Wide Area Network and blockchain to secure the network. Devices that expand coverage are paid in HNT tokens.	While main consensus is called Proof of Coverage, validation of coverage involves Federated Byzantine Agreement and Proof of Elapsed Time.	Provides an affordable and cost-effective solution for IoT device connectivity. Provides a secure and transparent platform to exchange data between one another.
Atonomi [13]: Built atop Ethereum. Uses a reputation based system to verify the identities and behavior of IoT devices. Under its ecosystems, devices from different manufacturers can seamlessly inter-operate.	Uses Proof of Reputation (PoR) which combines traditional Proof of Work Proof of Stake consensus mechanisms to achieve high efficiency	High scalability stems from PoR consensus. Only nodes with high reputation scores get to take part in consensus gathering. Devices are secured at onboarding.

Finally, no Web3 protocol is complete without a mature *token economics* model, one that will keep centralization concerns at bay, while ensuring a positive sum game for the facilitators.

In this paper, we are introducing a consensus protocol named Proof of Real-time Transfer (PoRT) with multiple revisions made to its prior version. We are also laying out our vision for an all encompassing Blockchain architecture that will help transitioning IoT services to Web3 one-step at a time.

II. LITERATURE REVIEW

Decentralized products and services in the spaces of finance, investments, gaming and collectibles grew in adoption over the past decade. The consensus protocols [1], [2], [3], [4] that power the layer 1 platforms serving such applications, each carefully selects an operating point that balances the needs for scale, security and decentralization and wherever applicable, layer 2 solutions [5] have ably bridged scalability gaps albeit at the transaction volumes that are far lower than that of IoT applications.

Many consensus protocols have been proposed for the onboarding of IoT applications onto Web3 platforms in the form of public blockchains and private blockchains. Common themes to such protocols has been that they are built for fast consensus, high throughput and low latency. Directed Acyclic Graph (DAG) based data structure was chosen over the traditional blockchain by a handful. Some protocols take energy expended in gathering consensus as a key design criteria. Table I summarizes some of the prominent Web3 protocols proposed for IoT systems.

With IoT devices not directly connected to blockchain platforms, but rather through entities that provide them with the requisite computing resources, BIoT paradigm [14] proposes protocols for the secure publication of sensor data in public blockchains. [15] proposes metrics to evaluate the efficiency of Blockchain-IoT architectures and propose a novel architecture named BIIT. They also delve into data storage challenges commonly observed in this sector. Blockchain driven access control protocols for smart grid systems is yet another popular research topic. [16], [17] propose well received solutions in this space. Data analytics and analytics driven inference are inseparable from the IoT ecosystem. Privacy preserving federated learning solutions [18], [19] have been proposed for developing analytics models through secure data sharing.

III. Proof of Real-time Transfer (PoRT)

A. Decentralized Identity and Device Onboarding

Our design begins with assigning decentralized identities to every entity (nodes, IoT devices, end users) that is either an inherent part of the ecosystem or one that interacts with it. Decentralized identities (DID) [21] plays a critical role in authentication, authorization and access control. Each entity generates a public-private key pair through elliptic curve digital signature algorithm such as Ed25519, derives its decentralized identify from a hash of the public key and submits a signed DID document comprising its public key, authentication mechanisms and service endpoints to the DID registry. It is estimated that generating 256 bit keys using Ed25519 requires 32 bytes of memory and the device needs to expend 450K operations in compute.

By design, a good proportion of IoT devices are constrained for memory and compute and will likely lack the capabilities required for key-pair generation and digital signatures. Onboarding them directly onto the proposed platform may be infeasible. A set of coexisting IoT devices typically communicate to the outside world through a gateway device (a central hub) through wired or wireless means (Wi-Fi, Bluetooth, Zigbee or Z-wave) using light-weight pub/sub protocols such as MQTT [20]. The gateway device is typically configured reasonably for compute, storage, power and network connectivity and serves as a liaison between the end-user and the respective devices. In similar vein, from a Web3 onboarding perspective, the gateway device shall serve as the Web3 point-of-contact, relaying sensor measurements adhering to protocols identified by the platform.

B. Peer-to-peer Networking Protocol

The proposed platform adopts the d-overlay network architecture [22] where each node connects with d peers and establishes direct port based communication with its peer. This design is resilient to network failure and facilitates decentralized censorship-resistant communication. With the primary purpose of the platform being decentralized delivery, storage and retrieval of IoT data streams, optimal selection of the d peers in the overlay network is critical as that plays a direct role in network throughput, network latency, resource lookup, data storage and its subsequent retrieval. PoRT adopts a combination of Perigee [23] and Kademlia protocols [24] in selecting the d best peers.

Perigee selects d-peers for each node of the network with the objective of minimizing overall network latency and optimal load balancing among the nodes of the d-overlay network. On the other hand, Kademlia facilitates efficient resource look-up across the network using Distributed Hash Tables (DHTs), playing a direct role in storage replication, storage tracking and data retrieval. Kademlia protocol is summarized below:

- Bitwise XOR distance: Every entity is associated with a unique 256 bit key (the same as DID) and distance between two keys is computed using bitwise XOR.

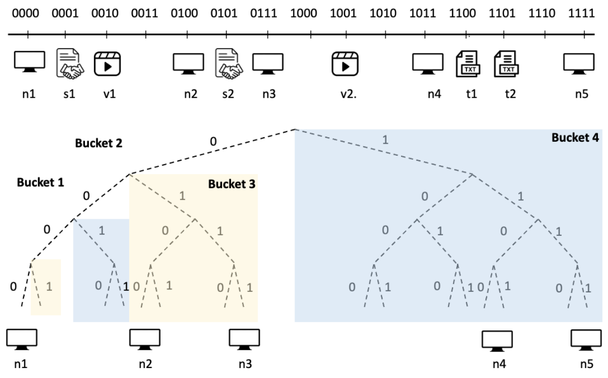

Fig. 1. An illustration of Kademlia 4 bit keyspace and the routing table of node n1. In the illustration {n1, n2, n3, n4, n5} correspond to nodes; {s1, s2} correspond to smart contracts; {v1, v2} correspond to videos; {t1, t2} correspond to text files

- Self-organizing of nodes in keyspace: With this distance metric, nodes organize themselves in the keyspace and are deemed as the point-of-contact for any $key, value$ pair that was closest to it in the keyspace.
- Routing table: Nodes maintain a routing table that assumes a tree-like structure, with every level representing a prefix of the binary node ID. The routing table is divided into buckets, with each bucket comprising nodes that share a common prefix and nodes look for peers that are closest to it in the keyspace.
- Resource lookup: To find a particular node or data from within the network, a node contacts nodes that are closer to the target ID in the keyspace and gradually narrows down the search space until the resource is found.
- Distributed Hash Table: DHTs are formed through in-part or holistic aggregation of continuously updated routing tables from across nodes.
- Bootstrap the network: For a node to join the network, the node must know the IP address and port number of at least one node of the network. The bootstrap node gathers information about other nodes of the network from its peers.

Figure 1 provides an illustration of a 4 bit keyspace, the routing table of a node and its set of buckets. In our design, node $n1$ shall be the point-of-contact for smart contract $s1$ and video $v1$, while node $n4$ shall the point-of-contact for text files $t1$ and $t2$. This association arises implicitly from keyspace proximity. PoRT protocol adopts a hybrid lookup algorithm combining the criteria set by Perigee and Kademlia in peer selection. PoRT currently adopts the tried and tested TCP/HTTP2.0 for network communication. A potential switch to the faster and more secure QUIC protocol is on the cards.

C. Standardized Message Schema

PoRT prescribes a standardized message schema for all devices onboarded onto the platform to adopt to send or receive messages. The standardized schema facilitates interoperability between devices and simplifies the network's message verification operations. Non-conforming messages are rejected by

TABLE II
STANDARDIZED DATA SCHEMA

Header	
Message ID	A globally unique identifier (GUID)
From address	DID URI of the producer
To address	DID URI of the consumer or group or self
Nonce	Number incremented for every new message
Schema	IoT-v-0.3
Timestamp	Dispatch timestamp
Time-to-live	{None, 1 week, 1 month, 1 year, 5 years}
Producer ACK	ACK = 0 or 1
Signature	Digital signature
Body	
Data	Encrypted payload

the platform. Decentralized Public Key Infrastructure (DPKI) plays a critical role in the message creation process. Let PbK_C and PvK_P be the public key of the consumer and the private key of the producer respectively. PoRT protocol encrypts payload at source and adopts digital signature scheme to guarantee data privacy, data integrity and authenticity, as described below.

$$Sign(Hash(Encrypt(data, PbK_C)), PvK_P) \tag{1}$$

Producers can request for any one of three services namely,

- Delivery only: Producer can set *Time-to-live* field to *None* and thereby request message delivery and not storage. The consumer's DID document should have listed the producer as an authorized sender of messages.
- Storage only: Producer can set *To address* to his own address and thereby request storage for the duration listed in the *Time-to-live* field.
- Delivery and Storage: Producer can opt for both services.

When ACK is set to 0, producer requests no acknowledgement of message delivery. When set to 1, producer shall receive acknowledgement when the designated leader received the message. Table II illustrates PoRT's standardized data schema.

D. Message Dispatch for Verification

The proposed platform makes available Network Discovery Service APIs that return the list of active nodes, their IP addresses and the respective ports open for communication. The producer dispatches the message created using the schema detailed above to 5 randomly selected nodes. The nodes run message verification function F_v as detailed in Algorithm 1. Message verification comprises a set of atomic operations that entail (a) signature verification (source authentication) (b) a check on data privacy and integrity (c) source authorization. It is computationally light and is highly parallelizable.

E. Dispatch to Leader Node for Delivery and Storage

If the message passes verification, then it dispatched to the *leader node* for fulfillment of delivery and storage as requested by the producer. Identifying the leader node involves no leader election. The leader node for a verified message is

Algorithm 1 Verification Function F_v

```
1: function VERIFY_MESSAGE(msg)
2:     if not $validSource$(msg.from) then
           return false
3:
4:     if not $validTarget$(msg.to) then
           return false
5:
6:     if not $authorizedTarget$(msg.to) then
           return false
7:
8:     if not $verifySignature$(signature) then
           return false
9:
   return true
```

none other than the node that is closest to the producer of that message in the keyspace K. The design of largely assigning a producer's messages to a leader was chosen to better assist delivery tracking and storage tracking for a producer across time. Dispatching the message to the leader node can happen either directly through port-based communication, if the leader node happens to be on the routing table or through Kademlia's efficient resource lookup approach. Let there be N nodes in the network and let the keys of n nodes in the network be N_i, $i \in (1..n)$. Let the producer's key be P. Let f correspond to the bitwise XOR distance function. The index of the leader node N_j^* is identified as

$$j^* = \arg\min_{i \in (1..n)} f(N_i, P) \tag{2}$$

If the message was marked up for delivery, the leader node N_j^* dispatches the message to the application layer for its timely delivery and . If the leader node N_j^* was not reachable, then the node that is next closest to the producer in K assumes responsibility for the message. The 5 nodes that were tasked with message verification are invested in the process until the leader node has officially dispatched the message for delivery. PoRT protocol decouples delivery of verified messages from fulfilment of storage requests, as the latter is not a real-time requirement as is the former.

F. PoRT Storage Thesis

On-chain storage, in the traditional sense, typically involves a leader who proposes a block, a committee of validators who arrive at a consensus in decentralized fashion on the validity of the block and the entire network working in unison towards maintaining a single universal truth on chain state. In the event of a fork, the process stalls until a corrective course of action is adopted. While this process imparts immutability to data stored on chain and embraces the core tenets of decentralization, the process can be slow and quite expensive for end users. While maintaining universal state is paramount for general DeFi applications, the question really is, is universal state maintenance a required feature for storing general purpose data in decentralized settings ? For instance, storing the image file of Bored Aped Yacht Club #3368 (133 KB) on Bitcoin and Ethereum will cost a prohibitive 0.0284 BTC and 7.9 ETH respectively [25]. Some of the popular standalone decentralized storage solutions such as [26], [27], [28] pose amenable costs for storing general purpose data.

PoRT proposes a fundamentally new solution for decentralized storage that comes with data immutability, data replication, storage tracking, persistence in data storage and data retrieval. PoRT's thesis for decentralized storage is that *locally maintained chains with partial state replication, the settings of which are entirely determined by verifiable random functions and cryptographic sortition, yields censorship resistance and preserves chain integrity while keeping centralization at bay*. In addition, PoRT adopts sortition driven behavioral audits, on-demand proofs of storage and slashing of incentives in the event of non-compliance. This disincentivizes maliciousness and promotes protocol adherence.

G. Three Local Chains

Each node maintains three local chains namely,

- Main chain: Comprises 4 temporal chains, each dedicated for data that required a certain time-lo-live (TTL) as illustrated in Table II. The node aggregates messages that it served as the leader for, sorts them into 4 blocks based on TTL, and adds the blocks to the corresponding temporal chain once it met a certain size criterion. Temporal chains are roll-over chains that retire the genesis block after a preset time has elapsed.
- Log chain: Summary of tasks performed by a node in serving the network: message verification, message delivery, storage, chain custodianship, behavioral audits are written onto blocks and saved onto this chain.
- Custody chain: Comprises snapshots of main chains, log chains and custodian chains from across the network apart from blocks that were sent to the node for storage by leader nodes for the respective blocks.

H. VRF and Cryptographic Sortition in Storage Replication

The standardized message schema illustrated in Table II implicitly introduces entropy in message creation. With fields such as timestamp, nonce, message ID being part of the message schema, messages sent to the network by different producers provide sufficiently as stores of entropy that the network can tap into to derive randomness in the system. Node N_j creates a block B comprising messages that the node is serving as a leader for. As mentioned above, all the messages comprised in block B have the same TTL factor. Using block B as a source of entropy, N_j generates a random number as described below. Let $(PbK_j,\ PvK_j)$ correspond to the public and private keys of N_j. Let h be the $SHA256$ hashing function and $<\ .\ >$ be the digital signature function. N_j generates a random number r_j as follows.

$$r_j = h(<(h(B),\ PvK_j>) \tag{3}$$

If other nodes of the network are provided with the random number r_j, the block B and the proof $\pi_j =< h(B),\ PvK_j >$, they can verify the *randomness* of r_j by checking if $<\pi_j,\ PbK_j>$ is equal to $h(B)$ and if $h(\pi_j)$ is equal to r_j. It is important to note that other nodes cannot guess r_j prior to N_j providing them with it, as generating r_j involves PvK_j that only N_j has in possession. PoRT combines the verifiability in the random number generated with cryptographic sortition in creating a dynamically evolving game of storage and audits, detailed below.

Let r_0, r_1, r_2, ..., r_{31} be the ordered set of bytes that make the random number r_j. Let $\oplus$ be the bitwise XOR operator. N_j maps r_j to a 8 bit number as shown below.

$$\hat{r}_j = (r_0 \oplus r_1 \oplus ... \oplus r_{31}) \tag{4}$$

N_j identifies the number of nodes it shall request dispatch block B to, requesting storage on their respective custody chains as $p = \max(5,\ \hat{r}_j \mod 10)$, where $\mod$ is the modulo operator. Next, N_j re-purposes the same random number r_j to identify the node indices of the p nodes. Let $>>$ correspond to the right bit shift operator. The p node indices are generated from $\hat{r}_j$ by applying right bit shift operation followed by a mod operation p times as $\hat{r}_j >> q \mod n$, where $q \in [1..p]$. Let N_i be a node index thus derived. N_j dispatches $\{B,\ r_j,\ \pi_j\}$ to N_i, requesting storage for B in N_i's custodian chain. N_j logs the retraceable steps above onto its log chain

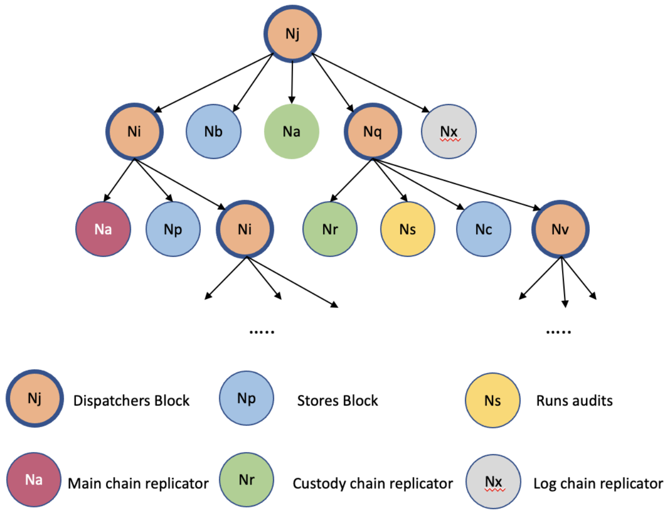

Fig. 2. An illustration of PoRT's game of storage and audits using cryptographic sortition

N_i verifies if r_j was indeed random and was obtained from block B. It further verifies if its node index fell within the list of indices that N_j was supposed to dispatch block B to, requesting storage. Upon passing the two checks, N_i generates its own random number r_i as $r_i = h(<(h(B),\ PvK_i>)$, where PvK_i is its private key. It maps r_i to a number in range (0,1), obtaining $\hat{r}_i$ (as illustrated for $\hat{r}_j$ in equation 4) and further dividing it by 255. PoRT protocol assigns preset ranges for the normalized $\hat{r}_i$ that each dictate a role that N_i is allowed to assume.

$$\begin{array}{rcl}
0 \leq \hat{r}_i < 0.3 & \Longrightarrow & \text{store block } B \text{ on custodian chain} \\
0.3 \leq \hat{r}_i < 0.4 & \Longrightarrow & \text{propagate } B \text{ to other nodes} \\
0.4 \leq \hat{r}_i < 0.6 & \Longrightarrow & \text{store } b \text{ blocks prior to B from } N_j\text{'s} \\
 & & \text{main chain onto its custodian chain} \\
0.6 \leq \hat{r}_i < 0.8 & \Longrightarrow & \text{store most recent } b \text{ blocks from } N_j\text{'s} \\
 & & \text{log chain onto its custodian chain} \\
0.8 \leq \hat{r}_i < 0.9 & \Longrightarrow & \text{store most recent } b \text{ blocks from } N_j\text{'s} \\
 & & \text{custodian chain onto its own chain} \\
0.9 \leq \hat{r}_i < 1 & \Longrightarrow & \text{run storage audits}
\end{array}$$

I. Running Storage Audits

Storage audits serve the key purpose of verifying if the node under audit has continuously maintained the integrity of the three local chains, with never once tampering with a chain apart from retiring genesis blocks when its time to expiry was met. Let N_x be the node performing the storage audit and let N_y be the node that is under audit. N_x gathers logs from its peers that pertains to their interactions with N_y and pieces together the random snapshots the network has saved from N_y's main chain, log chain and custodian chain. N_x audits N_y as detailed below

- Hash request based on chain height: Providing N_y with a set of ranges of block heights, N_x can request the hash for each range of blocks.
- Hash request based on byte ranges: Instead of hashes extracted using holistic blocks from the chain, N_y could request N_x for the hash of a certain range of blocks starting from byte b_1 until byte b_2.
- Block request: N_y could ask N_x to dispatch a specific block from a specific local chain.
- Age of blocks on temporal chains : N_y could ask N_x for the age of certain blocks on its temporal chains.

IV. Discussions and Conclusions

PoRT protocol preserves privacy of every entity in its ecosystem, secures data at source, preserves data integrity and authenticates every communication. The protocol adopts efficient peer-to-peer network design that combines Perigee and Kademlia in weighted fashion for low latency and efficient resource lookup. Its storage thesis is unique and sets a new paradigm for decentralized storage and storage audits. By applying cryptographic sortition, PoRT preserves role secrecy and mitigates many malicious attacks.

In future, we will expand on PoRT data retrieval, introduce Zero Knowledge proofs for storage, lay the foundations for executing smart contracts and analytics on our platform. We will formally introduce PoRT Tokenomics and will run extensive benchmarking exercises on the protocol.

References

[1] Nakamoto, Satoshi. “Bitcoin: A Peer-to-Peer Electronic Cash System.” (2008).

[2] Buterin, Vitalik. “A Next Generation Smart Contract & Decentralized Application Platform.” (2015).

[3] Yakovenko, A. T.. “Solana : A new architecture for a high performance blockchain v 0 . 8.” (2018).

[4] Chen, Jing and Silvio Micali. “Algorand: A secure and efficient distributed ledger.” Theor. Comput. Sci. 777 (2019): 155-183.

[5] Gangwal, Ankit, Haripriya Ravali Gangavalli and Apoorva Thirupathi. “A Survey of Layer-Two Blockchain Protocols.” J. Netw. Comput. Appl. 209 (2022): 103539.

[6] Popov, Serguei Yu.. “The Tangle.” (2015).

[7] Müller, Sebastian, Andreas Penzkofer, Nikita Polyanskii, Jonas Theis, William Sanders and Hans Moog. “Tangle 2.0 Leaderless Nakamoto Consensus on the Heaviest DAG.” IEEE Access 10 (2022): 105807-105842.

[8] IoTChain Team, “The IoTChain Whitepaper”, https://iotchain.io/pdf/web/ITCWHITEPAPER.pdf

[9] IoTeX Team, “Roll-DPoS: A Randomized Delegated Proof of Stake Scheme for Scalable Blockchain-Based Internet of Things Systems.” (2018).

[10] VeChain Team, “Web3 for better: Whitepaper 3.0”, https://www.vechain.org/assets/whitepaper/whitepaper-3-0.pdf

[11] The Hyperledger White Paper Working Group. “An Introduction to HyperLedger”, https://www.hyperledger.org/wp-content/uploads/2018/08/HL_Whitepaper_IntroductiontoHyperledger.pdf

[12] Haleem A, Allen A, Thompson A, Nijdam M, Garg R. “Helium: A Decentralized Wireless Network”, Release 0.4.2, 2018

[13] Emery V, Zamovsky A, Fragale D, Kinnaird P. “Atonomi: The Secure Ledger of Things v0.9.2”

[14] Urien, Pascal. “Blockchain IoT (BIoT): A New Direction for Solving Internet of Things Security and Trust Issues.” 2018 3rd Cloudification of the Internet of Things (CIoT) (2018): 1-4.

[15] Niya, Sina Rafati and Burkhard Stiller. “Efficient Designs for Practical Blockchain-IoT Integration.” NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium (2022): 1-6.

[16] Ding, Sheng, Jin Cao, Chen Li, Kai Fan and Hui Li. “A Novel Attribute-Based Access Control Scheme Using Blockchain for IoT.” IEEE Access 7 (2019): 38431-38441.

[17] Pinno, Otto Julio Ahlert, André Ricardo Abed Grégio and Luis Carlos Erpen De Bona. “ControlChain: Blockchain as a Central Enabler for Access Control Authorizations in the IoT.” GLOBECOM 2017 - 2017 IEEE Global Communications Conference (2017): 1-6.

[18] Lu, Yunlong, Xiaohong Huang, Yueyue Dai, Sabita Maharjan and Yan Zhang. “Blockchain and Federated Learning for Privacy-Preserved Data Sharing in Industrial IoT.” IEEE Transactions on Industrial Informatics 16 (2020): 4177-4186.

[19] Zhao, Yang, Jun Zhao, Linshan Jiang, Rui Tan, Dusit Tao Niyato, Zengxiang Li, L. Lyu and Yingbo Liu. “Privacy-Preserving Blockchain-Based Federated Learning for IoT Devices.” IEEE Internet of Things Journal 8 (2019): 1817-1829.

[20] Naik, Nitin. “Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP.” 2017 IEEE International Systems Engineering Symposium (ISSE) (2017): 1-7.

[21] Fdhila, Walid, Nicholas Stifter, Kristian Kost’al, Cihan Saglam and Markus Sabadello. “Methods for Decentralized Identities: Evaluation and Insights.” International Conference on Business Process Management (2021).

[22] Lua, Eng Keong, Jon A. Crowcroft, Marcelo Rita Pias, Ravi Sharma and Steven Lim. “A survey and comparison of peer-to-peer overlay network schemes.” IEEE Communications Surveys & Tutorials 7 (2005): 72-93.

[23] Mao, Yifan, Soubhik Deb, Shaileshh Bojja Venkatakrishnan, Sreeram Kannan and Kannan Srinivasan. “Perigee: Efficient Peer-to-Peer Network Design for Blockchains.” Proceedings of the 39th Symposium on Principles of Distributed Computing (2020):

[24] Maymounkov, Petar and David Mazières. “Kademlia: A Peer-to-Peer Information System Based on the XOR Metric.” International Workshop on Peer-to-Peer Systems (2002).

[25] @0xPhilillan and @FundamentalLabs, “Decentralized Storage: A Pillar of Web3”, June 2022.

[26] Protocol Labs, ”Filecoin: A Decentralized Storage Network”, 2017.

[27] Williams, Sam A., Viktor Diordiiev and Lev Berman. “Arweave: A Protocol for Economically Sustainable Information Permanence.” (2019).

[28] Wilkinson, Shawn. “Storj A Peer-to-Peer Cloud Storage Network.” (2014).