Blockchain Genesis (White-Paper)

Ing. Ivan Homoliak, Ph.D.
GentleBIT s.r.o. and
FIT, Brno Univeristy of Technology
i.homoliak@gentlebit.cz

Ing. Tomas Smetka
GentleBIT s.r.o. and
FIT, Brno Univeristy of Technology
t.smetka@gentlebit.cz

Ing. Jan Lochman
GentleBIT s.r.o. and
FNSPE, CTU in Prague
j.lochman@gentlebit.cz

***Abstract*—In this white-paper, we describe GEN Blockchain, the blockchain solution based on a public permissioned model and supporting Turing-complete smart contracts. GEN Blockchain is based on Hyperledger Besu but in contrast to its vanilla flavor, GEN Blockchain introduces a novel monetary policy model that regulates the block reward according to the current circulating supply.**

GEN Blockchain also contains its dedicated means for decentralized identity management, enabling any smart contract to perform KYC checks upon its discretion. Next, GEN Blockchain contains a reputation system for assessment of validators, which comes into play when the set of validators needs to be changed – therefore, we motivate the validators to behave honestly and provide high availability of their blockchain-related services. GEN Blockchain is intended to promote the adoption of decentralized applications in the region of Czech Republic.

I. Introduction

The popularity of blockchain systems has rapidly increased in the last decade, mainly due to the decentralization of control, immutability, and other features that blockchains provide. Cryptocurrency is the first practical application of blockchains, mostly known due to its first public permissionless deployment, Bitcoin [1]. In such a model, anyone can participate without revealing her identity, and potentially earn crypto-tokens by solving a Proof-of-Work (PoW) challenge using a dedicated mining hardware. Since the introduction of Bitcoin many practical decentralized use cases have emerged, such as asset tokenization, auctions, identity management, notaries, elections, etc. Nevertheless, such use cases require decentralized execution platforms supporting Turing-complete smart contract languages.

Ethereum [2] is the first public smart contract platform capable of handling decentralized applications. Unfortunately, the original idea of an arbitrary code execution in Ethereum became very limited due to low transaction throughput and high processing costs of Ethereum, which is unacceptable for many practical use cases that became prohibitively expensive and still not receive required finality. As a result, the permissioned blockchains based on Byzantine Fault Tolerant (BFT) consensus protocols started to gain adoption. In such permissioned blockchains, e.g., Hyperledger [3], only a limited number of known verified participants is operating, while these blockchains can provide high transaction throughput and almost immediate block finality.

A. Contributions

To promote the adoption of various blockchain-based applications in Czech Republic, we build GEN Blockchain, which is a public permissioned ledger with Turing-complete smart contract platform. The main aspect of GEN Blockchain is its high transaction throughput, (almost) immediate finality, and low operational costs for end-users. Moreover, to be compliant with KYC & AML practices, we incorporate decentralized identity management of end-users while we let the users creating smart contracts (for various businesses and use cases) to optionally choose polling of our identity registry available as a dedicated smart contract. Next, we introduce a novel monetary policy that automatically regulates the block reward according to the ratio of circulating supply to total supply.

II. Background

In this section, we describe background required to understand the remaining parts of the paper. Readers already familiar with principles of blockchains, smart contracts, and decentralized identity management can skip this section and proceed to Section III

A. Blockchain

Blockchain is ever-growing distributed ledger that is by design resistant to modifications. Blockchain consists of blocks that are linked using a cryptographic hash function, and each new block has to be agreed upon by participants running a consensus protocol (i.e., *consenus nodes* / *validators* / *participants*). Each block may contain orders transferring crypto-tokens, application codes written in a platform-supported language, and the execution orders of such applications. These application codes are referred to as *smart contracts* and can encode arbitrary processing logic (e.g., agreements). Interactions between clients and the cryptocurrency system are based on messages called *transactions*, which can contain either orders transferring crypto-tokens or calls of smart contract functions. All transactions sent to a blockchain are validated by validators who replicate the state of the blockchain.

B. Features

We summarize the most salient features of blockchains in the following [4].

Decentralization: is achieved by a distributed consensus protocol – the protocol ensures that each modification of the ledger is a result of interaction among participants. In the

consensus protocol, participants are equal, i.e., no single entity is designed as an authority. An important result of decentralization is resilience to node failures.

Censorship Resistance: is achieved due to decentralization, and it ensures that each valid transaction is processed and included in the blockchain.

Immutability: means that the history of the ledger cannot be easily modified – it requires a significant quorum of colluding nodes. The immutability of history is achieved by a cryptographic one-way function (i.e., a hash function) that creates integrity-preserving links between the previous record (i.e., block) and the current one. In this way, integrity-preserving chains (e.g., blockchains) or graphs (e.g., direct acyclic graphs [5]–[7] or trees [8]) are built in an append-only fashion. However, the immutability of new blocks is not immediate and depends on the time to the finality of a particular consensus protocol (see Section II-D).

Availability: although distributed ledgers are highly redundant in terms of data storage (i.e., full nodes store replicated data), the main advantage of such redundancy is paid off by the extremely high availability of the system. This feature may be of special interest to applications that cannot tolerate outages.

Auditability: correctness of each transaction and block recorded in the blockchain can be validated by any participating node, which is possible due to the publicly-known rules of a consensus protocol.

Transparency: the transactions stored in the blockchain as well as the actions of protocol participants are visible to other participants and in most cases even to the public.

C. Permission Models

Based on how a new node enters a consensus protocol, we distinguish the following blockchain types [4]:

Permissionless blockchains allow anyone to join the consensus protocol without permission. Such participation can be anonymous (or pseudonymous), and these protocols are designed to run over the Internet. To prevent Sybil attacks, this type of blockchains usually requires consensus nodes to establish their identifiers by running a Proof-of-Resource protocol, where the consensus power of a node is proportional to its resources allocated.

Permissioned blockchains require a consensus node to obtain permission to join the consensus protocol from a centralized or federated authority(ies), while nodes usually have equal consensus power (i.e., one vote per node). These schemes can be *public* if they are accessible over the Internet or *private* when they are deployed over a restricted network.

Semi-Permissionless blockchains require a consensus node to obtain some form of permission (i.e., stake) before joining the protocol; however, such permission can be given by any consensus node. The consensus power of a node is proportional to the stake that it has.

D. Consensus Protocols

There are several approaches to establishment of consensus among consensus nodes. In the following we describe the main types of consensus protocols:

Proof-of-Resource: a consensus nodes produces a valid block upon proving that the scarce resource was spent by a node to produce a valid block. For example in Proof-of-Work, nodes try to solve a puzzle which is often finding the hash value of a current block lower than some network-specified threshold (i.e., Nakamoto Consensus [1]). Another example are Proof-of-Storage protocols, where consensus nodes has to prove storing of certain data to produce a valid block.

Proof-of-Stake: in contrast to the previous category, no scarce resource is spent; instead, the nodes are required "to prove investment" of crypto-tokens to participate in a protocol, and thus eventually earn interest from the invested amount [9]. The advantage of Proof-of-Stake and Proof-of-Resource protocols is their high scalability in terms of number of consensus nodes, while they suffer from low transaction throughput and slow time to finality.[1]

Proof-of-Authority: these protocols were introduced as a possible response to a limited transaction throughput and slow time to finality in the previous two categories. In detail, the stake (of PoS protocols) is replaced by a reputation of a participant, while the number of participants is limited, and thus can utilize variants of BFT protocols that impose high overhead on message exchanges but ensure almost immediate finality.

E. Ethereum

Ethereum [2] is the first public permissionless smart contract platform using blockchain. The consensus protocol of Ethereum is based on *EtHash* Proof-of-Work but in contrast to Bitocoin's puzzle, it was designed to resists ASIC-based mining by requiring miners to store large portion of data with fast random access.

Each Ethereum consensus node contains Ethereum Virtual Machine (EVM) for execution of transactions that can deploy smart contract and execute their code. Such an execution of transactions modifies the global state of Ethereum, represented by the Merkle-Patricia trie (MPT), an integrity-preserving data structure, which contains data of all accounts states (simple accounts + smart contracts) and is maintained by every consensus node. Each account state contains the following:

- **nonce** – the number of transactions sent from the current account in the case of a simple account. In the case of a smart contract account, it represents the number of internally created contracts by the current smart contract. In both cases, this value serves as a reply attack protection, in which the attacker cannot re-execute the same transaction multiple times.
- **balance** – a balance of the account in Wei.

[1] I.e., one needs to wait several minutes until her transaction became irreversible with high enough probability.

- **storageRoot** – a root hash of the storage contents (i.e., the MPT trie) bound to the account state of a smart contract. If the account state is a simple account, this field is empty.
- **codeHash** – a hash value of the smart contract's code, which is executed when the smart contract receives a transaction with a call. If the account state is a simple account, this value is empty. Note that codes are stored in a dedicated key:value store.

*1) **Header**:* The Ethereum header contains the following:[2]

- **parentHash** – the hash of the previous block.
- **ommersHash** – the hash aggregating the uncle blocks (i.e., stale) blocks.
- **beneficiary** – the 160-bit long address of the consensus node who mined this block.
- **stateRoot** – the root hash of MPT representing the global state.
- **transactionsRoot** – the root hash of Merkle tree aggregating all transaction within the current block.
- **receiptsRoot** – the root hash of Merkle tree aggregating all execution receipts from EVM that correspond to all transactions within the block.
- **logsBloom** – the Bloom filter containing indexable data from logged entries – i.e., emitted events.
- **difficulty** – the difficulty value of PoW puzzle.
- **number** – the number of preceding blocks from the genesis block (i.e., height).
- **gasLimit** – a maximum gas capacity of the current block.
- **gasUsed** – a gas used by all transactions of the block.
- **timestamp** – a timestamp of the block creation.
- **extraData** – data related to EtHash PoW.
- **mixHash** – proof data related to EtHash PoW.
- **nonce** – proof data related to EtHash PoW.

*2) **Monetary Policy**:* Ethereum Foundation originally proposed fixed block reward of 5 ETH per block and no limit on total supply. However, by the time Ethereum decreased block reward two times (to 3 ETH in 2017 and to 2 ETH in 2019) through hard forks [10]. These two reward reductions happened as a part of Ethereum's philosophy of *minimum necessary issuance*. On top of the block reward, Ethereum pays up to 75% of the full block reward to miners of stale blocks (referred to as uncles or ommers), while it also incentivizes miners who include such blocks (in *ommersHash*).

F. Hyperledger Besu

Hyperledger Besu (further Besu) [11] is Java-based Ethereum client formerly known as Pantheon, which was developed mainly by PegaSys team from ConsenSys. Besu represents permissioned blockchain that can operate over public or private networks and it support account-based as well as node-based permissioning. The architecture of Besu is depicted in Figure 1, showing that most of the Besu components and principles are inherited from Ethereum.[3]

*1) **Standardization**:* Since Besu is based on principles of Ethereum, it implements the Enterprise Ethereum Alliance (EEA) standards for Ethereum-compatible permissioned blockchains [12] and Ethereum clients [13]. EEA envisions to standardize the consensus protocol, where the most likely candidate is IBFT 2.0.

*2) **Consensus Protocols**:* Besu enables to use consensus protocols based on PoW or PoA. PoA consensus protocols are specifically intended for consortium blockchains, and Besu currently supports two PoA consensus protocols [14]:

- **IBFT 2.0** – In IBFT 2.0, transactions and blocks are validated by pre-approved consensus nodes (a.k.a., validators). Validators create blocks in round robin fashion. Existing validators propose and vote to add or remove validators using the *extraData* header field. IBFT 2.0 has almost immediate finality. Hence, forks might barely occur and all valid blocks are included in the main chain.
- **Clique** – Clique is more fault-tolerant than IBFT 2.0. It tolerates up to $\frac{1}{2}$ of failing validators, while IBFT 2.0 tolerates only up to $\frac{1}{3}$ of failing validators. Clique does not have immediate finality, and thus forks might temporarily exist, which limits its utilization.

*3) **Permission Model**:* Besu provides two ways of *account permissioning* [15]: (1) **off-chain** (i.e., local) permissioning and (2) **on-chain** permissioning. Off-chain permissioning is individual to each validator, and enables validators to maintain their own white-list of addresses allowed to create transaction. In contrast, on-chain permissioning is common to all validators, which through their RPC service watch and parse the white-list stored in the account permissioning smart contract [16], while ignoring transactions coming from non-white-listed addresses. The white-list in smart contract is maintained by one or more Besu blockchain accounts with the admin role.

On top of account permissioning, Besu also provides *node permissioning*, and thus allows a single party to modify the list of validators according to her will, and thus degrades the decentralization of blockchains utilizing it.

*4) **Key Management**:* Besu supports standard Ethereum-compatible wallets for managing private keys, such as Metamask, MyEthetWallet, etc. ConsenSys also recommends EthSigner tool that provides access to key store (in a local file or cloud) and signs transactions via tools like Hashicorp Vault and Microsoft Azure.

*5) **API**:* User-oriented APIs of Besu conform to EEA [13] and thus support JSON-RPC and Ethereum's mainnet APIs. The APIs support WebSocket and HTTP protocols. Additionally to Ethereum, Besu also supports GraphQL API.

[2]Underlined fields are specific to PoW consensus protocol, and thus are mostly unrelated for this paper.

[3]Different components are IBFT and Clique consensus protocols as well as GraphQL API.

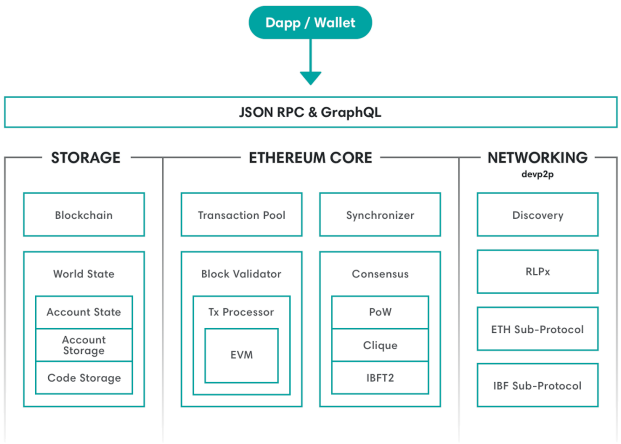

Figure 1. Architecture of Hyperledger Besu [14].

6) ***Header****:* To ensure compatibility with existing Ethereum development tools and wallets, Besu is using the same header structure as Ethereum. However, in the case of PoA setting (through IBFT 2.0), Besu does not utilize 4 PoW-specific header fields (i.e., underlined fields in Section II-E1 with exception of extraData). Nevertheless, these fields are part of the header data and contribute to the overall size of the blockchain.

G. Identity Management

Identity management refers to binding identities of entities to their public keys. This concept is also referred to as Public Key Infrastructure (PKI), and it has a few security goals [17]:

- **Accurate Registration:** The user must be unable to register an identity that she does not own.
- **Identity Retention:** The user must be unable to impersonate an identity already registered.
- **Censorship Resistance:** The user must be able to register any identity that she owns.

Using decentralized environment of blockchains can improve confidence in security goals of identity management by enabling users to own and manage their identity-related entries. Decentralized identity is addresses in W3C standard [18]

1) ***Decentralized ID (DID)****:* Decentralized Identifiers (DIDs) [18] represent a new type of universally unique identifiers whose control is decentralized since all roots of trust are contained in the blockchain and each entity might create its own root of trust (a.k.a., trusted anchor). DID employs the same hierarchical scheme for globally unique strings as URI, and it maps DID strings to DID documents containing data such as public keys, references to service endpoints associated with the entity, links to off-chain data, etc. Each DID document is stored in the blockchain and only the owner can create, manage, and prove ownership of her DID document.

2) ***ERC 725v1****:* is smart contract standard for managing identities of humans, groups, objects, and machines [19], in which, identity is associated with several keys serving various purposes, e.g., claims by individuals, quorums of keys to act on behalf of an organization's identity, access control, and privileges within an organization, etc. ERC 725v1 focuses on access control of smart contracts in proxy fashion, while each user requires her own contract. Moreover, all the keys are stored in the expensive storage of EVM.

3) ***ERC 1056****:* is a DID-compliant standard for manging identities with a limited use of blockchain resources [20] (i.e., storage in particular). Each identity can have unlimited number of *delegates* and *attributes* associated with it (representing DID document), while only the identifier of identity (DID) is stored in the expensive storage of EVM. Delegates and attributes of an identity are modified through emitting Ethereum events. This approach saves gas expenses imposed on EVM's storage but on the other hand requires a dedicated centralized service for processing of event stream and provisioning the most recent version of DID documents. In contrast to ERC 725v1 where each user requires her own contract, ERC 1056 serves as a single registry contract for all identities.

4) ***Identity-Oriented Blockchains****:* The Sovrin [21] is an example providing a public permissioned blockchain that consists of consensus nodes approved by Sovrin. Another example is Hyperledger Indy [22], which is project for a custom public permissioned blockchain intended for self-sovereign management of DIDs and their corresponding documents.

III. GEN Blockchain

In this section, we describe components of GEN Blockchain, its consensus mechanism, and monetary policy. We build GEN Blockchain as a public permissioned blockchain based on Hyperledger Besu, and thus it is fully compliant with EEA standards. Moreover, we maintain and offer to users the identity registry smart contract that provides a white-list of addresses in GEN Blockchain that have verified identities. However, it is an optional mechanism, and we do not limit utilization of GEN Blockchain by any account permissioning mechanisms. GEN Blockchain has its own native token denoted as `GEN`.

A. Consensus Protocol

Out of the consensus protocols provided by Besu, we opt for IBFT 2.0 since it best fits the needs of the PoA blockchain with a limited number of validators and requirements on high transactional throughput. Moreover, IBFT enables almost immediate finality, which on top of fast block creation rate (e.g., every 10s), offers almost real-time response of deployed DAPPs, so the users do not have to wait between any two dependent consecutive smart contract calls.

Due to high communication complexity of IBFT, the maximum number of validators participating in the consensus protocol is constrained to 50, and the minimum number of validators is equal to 4.

B. Identity Management

Inspired by the lightweight registry design of ERC 1056, we design a custom identity registry contract that maintains a white-list of accounts whose owners have verified identities

by an external identity provider. The list of identity providers is managed by a group of operators requiring a majority consensus on each action through multi-sig. We note that these identity providers as well as all users with verified identities exist on a dedicated Hyperledger Indy blockchain, and thus their public DID documents can be retrieved and verified. Therefore, our solution as a whole is DID-compliant.

1) ***Inserting Entries****:* To insert a new entry into identity registry, we do not require any special role and thus anybody who will submit a minimal verifiable credentials (MVCs) of some user with valid signature made by a known identity provider can create such an entry. MVCs contain the DID of the user's account in GEN Blockchain (i.e., encoded as the GEN address) and indication whether the user has legitimate identity (i.e., bool flag). Note that MVCs do expose only minimal private information about the users, which is very unlikely to cause any harm.

2) ***Revoking Entries****:* Each identity registry entry stores its creation date, and their are subject to *implicit* expiration after selected number of years[4]. After this time interval, the identities needs to be verified again. On top of implicit expiration, entries in identity registry can be revoked *explicitly* by a majority consensus of identity admins.

3) ***Utilizing Registry****:* To leverage the potential of our identity registry for KYC and AML purposes, users designing their smart contract can utilize an external method of our registry, which validates identity of any requested address, and thus provides access control.

C. Permission Model

We apply different permission models for validators and users, and we describe them in the following:

1) ***Validators****:* We follow a public *permissioned model* in the case of validators, in which a new validator can be added into GEN Blockchain upon approval of more than $\frac{2}{3}$ of existing validators. The procedure is executed on-chain, and Besu utilizes the *extraData* header field for this purpose.

2) ***Users/Clients****:* We follow a public *semi-permissionless* model for users, in which a new user account can be created by any existing user or validator by sending a non-zero amount of `GEN` to the account. Such an account can hold `GEN` and interact with any smart contract that do not enforce KYC by our identity registry contract Section III-B.

D. Reputation of Validators

Besu contains embedded mechanism for adding and removing validators from the consensus protocol. The corresponding data for this mechanism are part of the blockchain. To further facilitate the process of adding and removing validators and to motivate validators to act timely, a reputation system should reflect the *liveness* and *safety* as well as compliance.

[4]This number can be changed upon majority consensus of identity admins.

a) ***Liveness****:* The reputation system monitors the expected minted blocks versus truly minted valid blocks by particular validators. In similar way, we assess the reputation of validators based on availability during running of IBFT in the rounds when they are not selected as leaders. In this way, we can identify candidates for removal from GEN Blockchain and at the same time we motivate the validators to invest their resources into the quality of their network connection and availability of their hardware.

b) ***Safety****:* The reputation systems can also monitor invalid blocks minted by particular validators as well as minted blocks at the same height, with the intention to create the fork.

c) ***Compliance****:* Finally, the reputation system can reflect the compliance with agreed parameters such as minimum gas price. For example, one validator could decrease the minimum gas price and attract more client transactions, which might overall increase the obtained transaction fees. Another example from this category might assess the availability of validators within a dedicated off-chain distributed file system (DFS) if the blockchain platform assumes it. In this case, the availability proofs related to off-chain DFS can be provided and verified on-chain. The best candidate enabling provisioning of indisputable proofs of (non-)availability is Proof-of-Replication [23].

In the current version of GEN Blockchain, we incorporated *liveness-based* reputation assessment and we describe its details in the following. We note that other reputation assessment approaches will be part of the governance process.

1) ***Details of Current Reputation Assessment****:* Besu uses block header to store some information about validators, which can be accessed through Besu's RPC methods:

- $ibft\_getValidatorsByBlockNumber$ and
- $ibft\_getSignerMetrics$.

The former returns list of validators for given block number while the latter returns metrics showing how many blocks each validator validated in a given block range.

However, raw data returned by these methods are not sufficient to quickly assess reputation of validator. Therefore, GEN Blockchain is equipped with a reputation assessment application that regularly downloads data using the mentioned RPC methods and stores them locally. For any given range of blocks the application displays:

- A timeline showing percentage of blocks created by each validator (see Figure 2). If a validator omits some block creation in IBFT, this chart can reveal it.
- A timeline displaying active intervals of validation for each validator (see Figure 3).
- A table with basic statistics – the number of blocks each validator validated, should validate, and missed (see Figure 4).

E. Incentives and Monetary Policy

GEN Blockchain allows defining *treasury* (i.e., the improvement reserve) and *operator* addresses, which are used as recipients obtaining a given percentage of block reward and

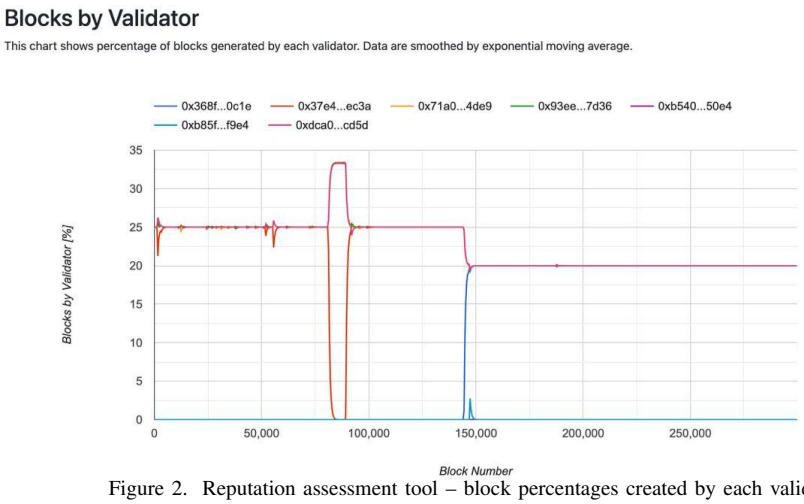

Figure 2. Reputation assessment tool – block percentages created by each validator.

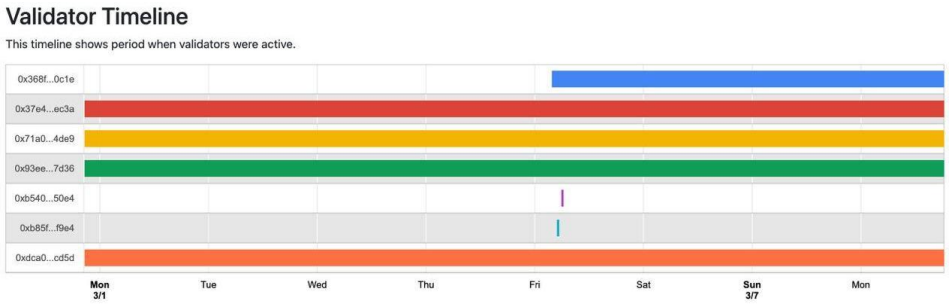

Figure 3. Reputation assessment tool – activity of validators.

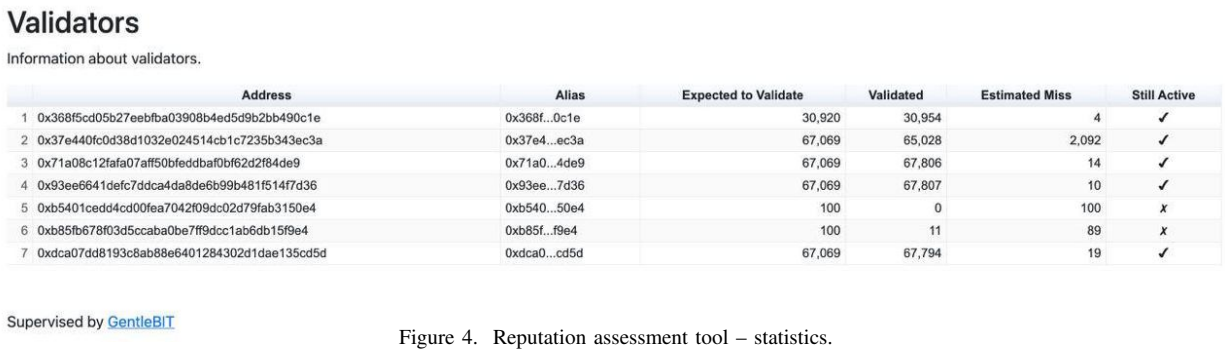

Figure 4. Reputation assessment tool – statistics.

block fees, while the validator of a block receives the rest. In the initial version of GEN Blockchain, *treasury* receives 5% of block reward and *operator* receives 10% of block fees.

Unlike other POA blockchains that use a fixed block reward, GEN Blockchain uses its own self-regulating block reward system, which adjusts block reward based on a total GEN supply distribution. We detail it in the following.

1) ***Reward Self-Regulation****:* On top of Besu's *epochlength* parameter, GEN Blockchain monetary system introduces additional epoch length parameter L, defining a block period after which the block reward is periodically adjusted by the network. Given a block number n, we define epoch number $\epsilon = \lfloor n/L \rfloor$.

GEN Blockchain adapts the *difficulty* field of the header to represent the block reward. We define the initial block reward r_0 and initial difficulty d_0 in the genesis file. The block reward r is then calculated from difficulty d as

$$r = r_0 \cdot \frac{d_0}{d}. \tag{1}$$

In epoch $\epsilon = 0$, all blocks have difficulty $d = d_0$ so all block rewards are equal to r_0. Entering the new epoch causes a change in difficulty d, which in turn changes the reward r. The difficulty remains unchanged during the epoch. Difficulty calculation uses PID regulation [24].

For a given block number, we define S_{total} as the total number of GEN minted, S_{val} as the number of GEN on all validators' addresses and circulating supply as $S_{circ} =$

$S_{total} - S_{val}$. Therefore, the ratio

$$\varphi = \frac{S_{total}}{S_{circ}} = 1 + \frac{S_{val}}{S_{circ}} \tag{2}$$

represents the fraction of all GEN on validators' addresses. When $\varphi \to 1$, percentage of GEN held by validators is close to zero. The higher the value of φ, the higher percentage of GEN is held by validators.

In our model, the value of φ should decrease in time, as S_{total} increases. It's because we want smaller fraction of GEN to be on addresses of validators as S_{total} increases. To express this mathematically, initial ratio φ_0, final ratio $\varphi_\infty < \varphi_0$ and ratio decay $\alpha \in (0; 1)$ are introduced in the genesis file. Target ratio $\varphi_t(\epsilon)$ is expressed by exponential decrease of φ from φ_0 to φ_∞ as a function of epoch number ϵ:

$$\varphi_t(\epsilon) = \varphi_\infty + (\varphi_0 - \varphi_\infty) \cdot \alpha^\epsilon. \tag{3}$$

The aim of the PID regulation is to modify block reward r in such a way that φ keeps very close to φ_t. To express how much φ_t differs from φ, we introduce regulation factor ψ. Both φ and φ_t are rounded down to 3 decimal places before entering the following expression:

$$\psi = \frac{\varphi - \varphi_t}{\sqrt{\max(\varphi_t - \varphi_\infty; 0.001)}}. \tag{4}$$

To cutoff extreme values of ψ, the values of $\psi > 1$ are saturated to 1 and values of $\psi < -1$ are saturated to -1. For further calculations, ψ is rounded down to 3 decimal places. The value of ψ can be interpreted in the following way;

- If $\psi > 0$, validators hold more GEN than desired. The bigger ψ value, the bigger the difference. The block reward r should decrease in the next epoch.
- If $\psi = 0$, validators hold exactly the number of GEN they should hold according to our model. The block reward should not change in the next epoch.
- If $\psi < 0$, validators hold less GEN than desired. The lower ψ value, the higher the difference. The block reward r should be increased in the next epoch.

Values of ψ enter PID regulation to calculate the ratio of a difficulty change. Suppose that GEN Blockchain contains the last block of epoch ϵ and is about to create the first block of epoch $\epsilon + 1$. Let ψ_ϵ to represent a value of ψ in the last block of epoch ϵ, $\psi_{\epsilon-1}$ value of ψ in the last block of epoch $\epsilon - 1$ and so on. Then we define

$$\Psi_\epsilon = \tanh\left(k_p\psi_\epsilon + k_i \sum_{i=0}^{P-1} \psi_{\epsilon-i} + k_d\left(\psi_\epsilon - \psi_{\epsilon-1}\right)\right), \tag{5}$$

where k_p, k_i and k_d are positive parameters of proportional, integral, and derivative part of PID regulation and P is a period over which integral part sums. All of these parameters are defined in the genesis file. The PID result is normalized by hyperbolic tangent ensuring that $\Psi_\epsilon \in [-1; 1]$. In the case of $\psi_{\epsilon-1}$ is not defined, derivative part is omitted.

The resulting Ψ_ϵ is rounded down to 3 decimal places and used for calculation of difficulty $d_{\epsilon+1}$ for epoch $\epsilon + 1$.

- $d_{\epsilon+1} = d_\epsilon$ if $\Psi_\epsilon = 0$,
- $d_{\epsilon+1} = d_\epsilon \cdot (1 + \Psi_\epsilon)$ if $\Psi_\epsilon > 0$,
- $d_{\epsilon+1} = d_\epsilon / (1 - \Psi_\epsilon)$ if $\Psi_\epsilon < 0$.

These calculations ensure that both difficulty and reward change in the range of $[-50\%; +100\%]$ from their previous values. Next, we define additional variables in the genesis file which define the minimal and maximal value of the difficulty, which are taken into account when modifying the difficulty.

Another parameter of the genesis file is the *minimal block reward*, which determines the maximal difficulty. In addition, we introduce *minimal inflation* parameter which enforces block reward to be greater then or equal to a value guaranteeing the total supply S_{total} to increase by *minimal inflation* in 365 days. The *minimal inflation* parameter is intended for the later epochs of GEN Blockchain, while direct definition of *minimal block reward* suits the first epochs. *Minimal difficulty* determines *Maximal block reward*, and it is included in the genesis file directly.

F. Governance

To reach a decision on particular problems and questions that might arise during the operation of GEN Blockchain, we specify the governance model and the topics it is dealing with.

*1) **Body**:* The governance body of GEN Blockchain consist of all validators and decisions are reached through smart contract voting using multi-sig. Since validators are rewarded for running the network and are accountable for their actions, they are expected to make decisions that are beneficial for the platform's long-term growth and stability.

We enable any validator to start a voting on a particular matter. The voting si currently publicly visible – everyone sees what options validators voted for. However, in some improvement proposal, we also plan to enable privacy preserving voting [25], [26], where the votes of validators are blinded by zero knowledge constructs.

*2) **Topics**:* The governance process can control the following three topics:

- **technology** – this topic involves technological updates of the platform, approval of new tool development, parameters such as block creation rate and gas limit per block, fixing bugs through hard forks, etc.
- **economy** – the monetary policy might be updated with the intention to regulate inflation rate, the size of the improvement reserve and purposes of its utilization.
- **governance** – this category covers changes to the governance model, for example adding end-users to specific kind of voting.

*3) **Improvement Reserve (Treasury)**:* Since technology is constantly improving, the ecosystem of GEN Blockchain needs an ability to quickly adapt on the technology changes, inventions, and breakdowns. To quickly respond on technological changes, we maintain an improvement reserve within

a dedicated smart contract, which is funded in each block with 5% of the block reward. The balance of the contract can be transfered through majority consensus of validators, which is achieved by publicly visible multi-sig control.

Besides the reaction on technological changes, this fund is also intended for maintenance of development and user-oriented tools underpinning GEN Blockchain, including blockchain explorers, reputation system, identity management, standard DAPPs, testing networks, etc.

G. Wallets

All existing Ethereum wallets are compatible with GEN Blockchain since Besu conforms to EEA standards (see Section II-F1). The important aspect of the wallet is the possibility of pairing with a hardware wallet, such as Trezor, Ledger, KeepKey, or Evoled. One notable example of open source wallet supporting such pairing is Metamask,[5] which comes in the form of the browser plugin.

*a) **Smart Contract Wallets with 2FA**:* For the higher amounts of `GEN`, the clients might create a smart contract wallet secured with two-factor authentication, such as *Gnosis* [27] or *SmartOTPs* [28].

In Gnosis, the client needs to have two private keys stored in – either two hardware wallets or one hardware wallet and one software wallet. Then, to make an operation with the wallet (i.e., transfer), the client creates and submits two transactions with the same operation but signed by different wallets.

In SmartOTPs, the client needs to posses one hardware wallet and one authenticator token (e.g. App in smart phone). To make an operation with the wallet, the client first submits a signed transaction with hardware wallet and then she submits an OTP obtained from the authenticator within the second transaction.

H. Recovery of Private Keys

Standard way of key recovery for any single factor wallet is through backed up mnemonic HD seed associated with the wallet. The users should store these seeds in a secure places.

We note that SmartOTPs [28] provide recovery for lost secrets (i.e., authenticator, hardware wallet, or both) through the last resort address and timeout functionality – after expiration of a timeout, any exiting user of GEN Blockchain can call a method of a smart contract, which will send the remaining balance to the last resort address.

I. Minimal Hardware Requirements for Validators

Validators in GEN Blockchain must be equipped at least with the following hardware:

- 4x CPU clocked at 2.4 GHz,
- 8 GB RAM, and
- 500 GB SSD hard drive.

In the case of using AWS EC2, we recommend to use at least the *c5.xlarge* instance and *compute optimized* option.

[5]https://metamask.io/

J. Faucet

GEN Blockchain Faucet consists of both smart contract and web application. The smart contract Faucet allows sending of 1 `GEN` to a given address and is limited to one withdraw per 10 blocks per address. Due to the smart contract nature of the faucet, `GEN` can be distributed even without the web application.

Web application part of the Faucet is equipped with CLI that enables deployment of smart contract Faucet and its initialization with some funds. Then, Web UI of the Faucet enables to call Faucet smart contract and thus send 1 GEN to a given GEN Blockchain address.

IV. Evaluation

In this section, we estimate the storage consumption imposed by using GEN Blockchain in terms of requirements on storing the blockchain itself and its global state.

*1) **Size of the Blockchain**:* It is important to estimate the size of GEN Blockchain in time. In the following, we assume the header size of 508B. If we were to create empty blocks with block creation time of 10s, then the blockchain would grow by ∼3.15MB every year (i.e., $\frac{3600}{10s} \times 24 \times 365$). Hence, the overhead for fast time to finality and a short block creation time is negligible.

If were to assume 100K simple payment transactions per day, GEN Blockchain would grow by 7.77GB every year (i.e., $10^5 \times |tx_p| \times 365 + 3.15MB$, where $|tx_p| = 203B$). In the case of the same number of smart contract calls assuming the average transaction size $|tx_s| = 500B$, the yearly grow of blockchain size would be 18.26GB (i.e., $10^5 \times |tx_s| \times 365 + 3,15MB$); however, one needs to account for smart contract deployments, where the code size of a single contract might be up to 512kB.

*2) **Size of the Global State**:* On top of the blockchain data, each validator has to maintain the global state consisting of all account state objects aggregated by Merkle-Patricia Trie (MPT). The size of the global state depends on three items:

- the number of accounts,
- the number of nodes in MPT trie,
- the size of storages associated with the accounts (in the case of smart contracts),
- the code size of accounts (in the case of smart contracts).

We demonstrate a size of the global state for several full MPT (containing the number of simple accounts equal to powers of 16) in Table I, where we assume that the size of each account state equal to 42B and size of a each branch node equal to 512B. Note that if we were to assume all the citizens of Czech Republic to be part of GEN Blockchain, we would end up on the sixth level of MPT, consuming up to 1.3GB of data.

The model example of the global state consisting of smart contract accounts only is showed in Table II, where we assume the average size of smart contract code equal to 4kB and the average storage size of each contract equal to 16kB. We note that smart contract account are in general no so frequent as simple accounts. Nevertheless, if we were assume all the

MPT Level	# of Simple Accounts	Size of Global State
1	16	1.18 kB
2	256	19.45 kB
3	4096	311.80 kB
4	65.55×10^3	4.99 MB
5	1.04×10^6	79.83 MB
6	16.77×10^6	1.27 GB
7	0.26×10^9	20.04 GB
8	4.29×10^9	326.99 GB
9	68.71×10^9	5.23 TB
10	1.09×10^{12}	83.71 TB

Table I
SPACE REQUIREMENTS ON GLOBAL STATE CONSISTING OF SIMPLE ACCOUNTS ONLY.

MPT Level	# of Smart Contracts	Size of Global State
1	16	18.55 kB
2	256	4.88 MB
3	4096	76.11 MB
4	65.55×10^3	1.21 GB
5	1.04×10^6	19.48 GB
6	16.77×10^6	311.79 GB

Table II
SPACE REQUIREMENTS ON GLOBAL STATE CONSISTING OF SMART CONTRACTS ONLY. WE ASSUME AVERAGE CODE SIZE OF SMART CONTRACT AND STORAGE EQUAL TO 2KB AND 16KB, RESPECTIVELY.

citizens of Czech Republic to have exactly one smart contract, then the size of the global state would be up to 320GB.

V. DISCLAIMER

The content of this document is subject to a change in accordance to the governance model proposed, laws, regulations, and advancements of the technology and research.

VI. CONCLUSION

In this white-paper, we described GEN Blockchain with its public permissioned model. GEN Blockchain is based on Hyperledger Besu but in contrast to its vanilla flavor, GEN Blockchain introduces a novel monetary policy model that regulates the block reward according to the current circulating supply. GEN Blockchain also contains its dedicated means for decentralized identity management, enabling any smart contract to perform KYC checks upon its discretion, and thus increase the reputation and credibility of the business or use case that it addresses. Next, we created a liveness-based reputation system for assessment of validators' performance and availability, which serves as an input for adjustments of validator set. In future work, we plan to incorporate new proposals and (mentioned) improvements as part of the governance model.

ACKNOWLEDGMENT

We would like to thank to XIXOIO a.s. for supporting the research and development of blockchain projects in Czech Republic. In particular, we would like to thank XIXOIO a.s. for the support and funding related to the white-paper of GEN Blockchain and its design.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] Ethereum team, "A Next-Generation Smart Contract and Decentralized Application Platform," https://github.com/ethereum/wiki/wiki/White-Paper#modified-ghost-implementation, 2018.

[3] Hyperledger Team, "Hyperledger architecture, volume 1: Consensus," 2017. [Online]. Available: https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf

[4] I. Homoliak, S. Venugopalan, D. Reijsbergen, Q. Hum, R. Schumi, and P. Szalachowski, "The security reference architecture for blockchains: Towards a standardized model for studying vulnerabilities, threats, and defenses," IEEE Communications Surveys & Tutorials, 2020.

[5] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "Spectre: A fast and scalable cryptocurrency protocol." IACR Cryptol. ePrint Arch., vol. 2016, p. 1159, 2016.

[6] Team Rocket, "Snowflake to avalanche: A novel metastable consensus protocol family for cryptocurrencies," 2018. [Online]. Available: https://www.semanticscholar.org/paper/Snowflake-to-Avalanche-%3A-A-Novel-Metastable-Family/85ec19594046bbcfe12137c7c2e3744677129820?p2df

[7] S. Popov, "The Tangle," http://tanglereport.com/wp-content/uploads/2018/01/IOTA_Whitepaper.pdf, 2016.

[8] Y. Sompolinsky and A. Zohar, "Accelerating bitcoin's transaction processing. fast money grows on trees, not chains." IACR Cryptology ePrint Archive, vol. 2013, no. 881, 2013.

[9] QuantumMechanic, "Proof of stake instead of proof of work," 2011. [Online]. Available: https://bitcointalk.org/index.php?topic=27787.0

[10] E. Foundation, "Monetary Policy," 2020. [Online]. Available: https://docs.ethhub.io/ethereum-basics/monetary-policy/

[11] Hyperledger Besu, "Besu Enterprise Ethereum Client," 2021. [Online]. Available: https://besu.hyperledger.org/en/stable/

[12] R. Coote, G. Polzer, and G. Noble, "Enterprise Ethereum Alliance Permissioned Blockchains Specification v2," 2020. [Online]. Available: https://entethalliance.org/wp-content/uploads/2020/11/EEA_Enterprise_Ethereum_Permissioned_Blockchains_Specification_v2.pdf

[13] C. Nevile, G. Polzer, R. Coote, G. Noble, D. Burnett, and D. Hyland-Wood, "Enterprise Ethereum Alliance Client Specification v6," 2020. [Online]. Available: https://entethalliance.org/wp-content/uploads/2020/11/EEA_Enterprise_Ethereum_Client_Specification_v6.pdf

[14] R. Dawson and M. Baxter, "Announcing Hyperledger Besu," 2017. [Online]. Available: https://www.hyperledger.org/blog/2019/08/29/announcing-hyperledger-besu

[15] H. Besu, "Permissioning," 2020. [Online]. Available: https://besu.hyperledger.org/en/stable/Concepts/Permissioning/Permissioning-Overview/

[16] ConsenSys, "Permissioning Smart Contracts."

[17] C. Fromknecht, D. Velicanu, and S. Yakoubov, "A decentralized public key infrastructure with identity retention." IACR Cryptology ePrint Archive, vol. 2014, p. 803, 2014.

[18] W3C community, "Decentralized Identifiers (DIDs)," 2019. [Online]. Available: https://w3c-ccg.github.io/did-spec/

[19] F. Vogelsteller and T. Yasaka, "ERC-725 Ethereum Identity Standard," 2017. [Online]. Available: https://github.com/ethereum/EIPs/blob/ede8c26a77eb1ac8fa2d01d8743a8cf259d8d45b/EIPS/eip-725.md

[20] P. Braendgaard and J. Torstensson, "ERC1056: Lightweight Identity," 2018. [Online]. Available: https://github.com/ethereum/EIPs/issues/1056

[21] A. Tobin and D. Reed, "The inevitable rise of self-sovereign identity," 2016. [Online]. Available: https://sovrin.org/wp-content/uploads/2018/03/The-Inevitable-Rise-of-Self-Sovereign-Identity.pdf

[22] Hyperledger Indy, "Introduction to Hyperledger Indy," 2019. [Online]. Available: https://github.com/hyperledger-archives/education/blob/master/LFS171x/docs/introduction-to-hyperledger-indy.md#internet-identity-with-hyperledger-indy

[23] J. Benet, D. Dalrymple, and N. Greco, "Proof of replication," Protocol Labs, July, vol. 27, 2017.

[24] K. H. Ang, G. Chong, and Y. Li, "Pid control system analysis, design, and technology," IEEE Transactions on Control Systems Technology, vol. 13, no. 4, pp. 559–576, 2005.

[25] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for board-room voting with maximum voter privacy," in International Conference on Financial Cryptography and Data Security. Springer, 2017, pp. 357–375.

[26] S. Venugopalan, I. Homoliak, Z. Li, and P. Szalachowski, "Bbb-voting: 1-out-of-k blockchain-based boardroom voting," arXiv preprint arXiv:2010.09112, 2020.

[27] ConsenSys, "Gnosis Wallet," 2019. [Online]. Available: https://github.com/Gnosis/MultiSigWallet

[28] I. Homoliak, D. Breitenbacher, O. Hujnak, P. Hartel, A. Binder, and P. Szalachowski, "SmartOTPs: An air-gapped 2-factor authentication for smart-contract wallets," in Proceedings of the 2nd ACM Conference on Advances in Financial Technologies, AFT 2020, New York, NY, USA, October 21-23, 2020. ACM, 2020. [Online]. Available: https://doi.org/10.1145/3419614.3423257