Innovation, Sciences et Développement économique Canada
Office de la Propriété Intellectuelle du Canada

Innovation, Science and Economic Development Canada
Canadian Intellectual Property Office

CA 3098766 A1 2020/01/16

(21) 3 098 766

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(86) Date de dépôt PCT/PCT Filing Date: 2019/10/15
(87) Date publication PCT/PCT Publication Date: 2020/01/16
(85) Entrée phase nationale/National Entry: 2020/10/29
(86) N° demande PCT/PCT Application No.: CN 2019/111316
(87) N° publication PCT/PCT Publication No.: 2020/011287

(51) Cl.Int./Int.Cl. *H04L 29/08* (2006.01)

(71) Demandeur/Applicant:
ALIPAY (HANGZHOU) INFORMATION TECHNOLOGY CO., LTD., CN

(72) Inventeur/Inventor:
ZHUO, HAIZHEN, CN

(74) Agent: KIRBY EADES GALE BAKER

(54) Titre : INDEXATION ET RECUPERATION DE DONNEES DE CHAINE DE BLOCS CODEES
(54) Title: INDEXING AND RECOVERING ENCODED BLOCKCHAIN DATA

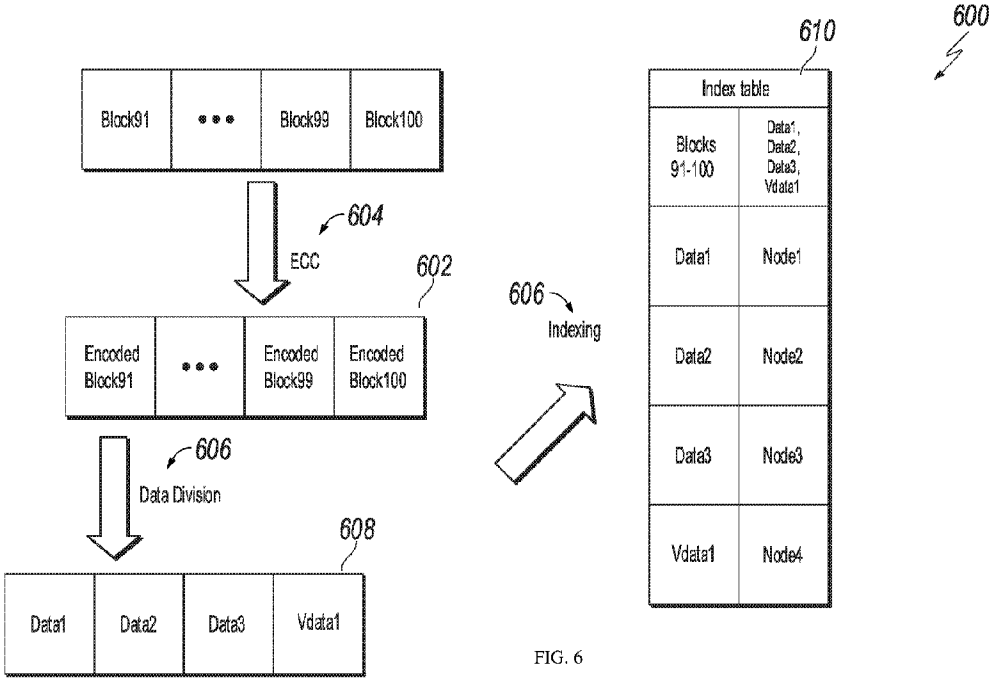


FIG. 6

(57) Abrégé/Abstract:
Disclosed herein are methods, systems, and apparatus, including computer programs encoded on computer storage media for indexing blockchain data for storage. One of the methods includes generating a plurality of encoded blocks based on performing error correction coding (ECC) on a plurality of blocks of a blockchain; for each encoded block of the plurality of encoded blocks: dividing the encoded block into a plurality of datasets based on a data storage scheme associated with the plurality of blocks, wherein the data storage scheme provides assignments of the plurality of datasets to a plurality of blockchain nodes; storing at least one of the plurality of datasets based on the assignments provided in the data storage scheme; and providing an index that indexes each of the plurality of datasets to each of the plurality of the blockchain nodes at which a respective dataset is stored.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
16 January 2020 (16.01.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/011287 A3

(51) International Patent Classification:
*H04L 29/08* (2006.01)

(21) International Application Number:
PCT/CN2019/111316

(22) International Filing Date:
15 October 2019 (15.10.2019)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **ALIPAY (HANGZHOU) INFORMATION TECHNOLOGY CO., LTD.** [CN/CN]; No. 556 Xixi Road, 8th Floor, Section B, Suite 801-11, West Lake District, Hanzhou, Zhejiang 310000 (CN).

(72) Inventor: **ZHUO, Haizhen**; No. 556 Xixi Road, 8th Floor, Section B, Suite 801-11, West Lake District, Hangzhou, Zhejiang 310000 (CN).

(74) Agent: **BEIJING BESTIPR INTELLECTUAL PROPERTY LAW CORPORATION**; Room 409, Tower B, Ka Wah Building, No. 9 Shangdi 3rd Street, Haidian District, Beijing 100085 (CN).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA,

(54) Title: INDEXING AND RECOVERING ENCODED BLOCKCHAIN DATA

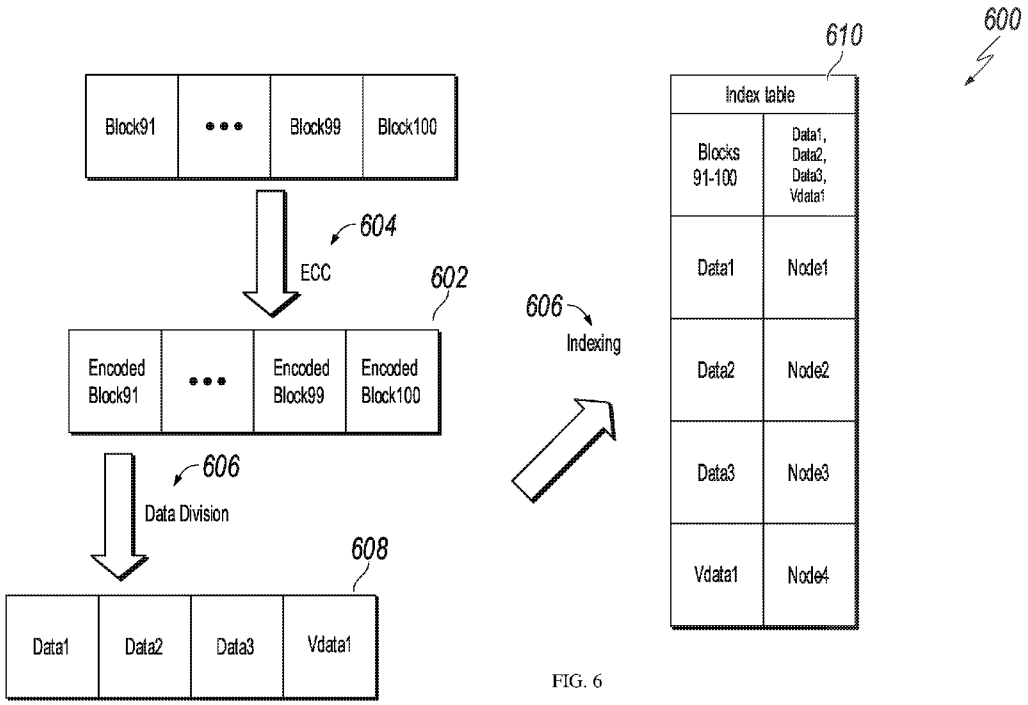


FIG. 6

(57) Abstract: Disclosed herein are methods, systems, and apparatus, including computer programs encoded on computer storage media for indexing blockchain data for storage. One of the methods includes generating a plurality of encoded blocks based on performing error correction coding (ECC) on a plurality of blocks of a blockchain; for each encoded block of the plurality of encoded blocks: dividing the encoded block into a plurality of datasets based on a data storage scheme associated with the plurality of blocks, wherein the data storage scheme provides assignments of the plurality of datasets to a plurality of blockchain nodes; storing at least one of the plurality of datasets based on the assignments provided in the data storage scheme; and providing an index that indexes each of the plurality of datasets to each of the plurality of the blockchain nodes at which a respective dataset is stored.

WO 2020/011287 A3

[Continued on next page]

WO 2020/011287 A3

SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

**Published:**

— *with international search report (Art. 21(3))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

— *upon request of the applicant, before the expiration of the time limit referred to in Article 21(2)(a)*

(88) **Date of publication of the international search report:**
20 August 2020 (20.08.2020)

# INDEXING AND RECOVERING ENCODED BLOCKCHAIN DATA

## TECHNICAL FIELD

**[0001]** This specification relates to indexing and recovering blockchain data encoded based on error correction coding.

## BACKGROUND

**[0002]** Distributed ledger systems (DLSs), which can also be referred to as consensus networks, and/or blockchain networks, enable participating entities to securely and immutably store data. DLSs are commonly referred to as blockchain networks without referencing any particular user case. Examples of types of blockchain networks can include public blockchain networks, private blockchain networks, and consortium blockchain networks. A consortium blockchain network is provided for a select group of entities, which control the consensus process, and includes an access control layer.

**[0003]** Blockchain-based programs can be executed by a distributed computing platform. For example, the distributed computing platform can include a virtual machine that provides the runtime environment for executing smart contracts. A blockchain computing platform can be viewed as a transaction-based state machine. State data in the platform can be assembled to a global shared-state referred to as a world state. The world state includes a mapping between account addresses and account states. The world state can be stored in data structures such as the Merkle Patricia tree (MPT).

**[0004]** Besides state data, blockchain networks can also store other types of data such as block data and index data. Block data can include block header and block body. The block header can include identity information of a particular block and the block body can include transactions that are confirmed with the block. As transactions are increasingly entered into the blockchain, state data and block data can grow very large in size. In some DLSs, every node stores an entire copy of the blockchain, which can take large amount of storage space. This is because all block data and state data are stored going back to the first transaction recorded to the blockchain. In some DLSs, a few shared nodes store the entire copy of the blockchain and share blockchain data with other blockchain nodes which can create "data inequality." That is, when data are unevenly distributed across different nodes, the risk of data security can be high when nodes that store majority of data are at fault.

[0005] Accordingly, it would be desirable to enable such storage in a manner that maintains data equality and data processing efficiency. It would also be desirable to enable storage of data on nodes in the DLS in a manner that reduces consumption of computational resources or memory, while being able to efficiently recover and retrieve the original data when needed.

## SUMMARY

[0006] Described embodiments of the subject matter can include one or more features, alone or in combination.

[0007] For example, a computer-implemented method is provided for indexing blockchain data for storage. The method comprises: generating a plurality of encoded blocks based on performing error correction coding (ECC) on a plurality of blocks of a blockchain; for each encoded block of the plurality of encoded blocks: dividing the encoded block into a plurality of datasets based on a data storage scheme associated with the plurality of blocks, wherein the data storage scheme provides assignments of the plurality of datasets to a plurality of blockchain nodes; storing at least one of the plurality of datasets based on the assignments provided in the data storage scheme; and providing an index that indexes each of the plurality of datasets to each of the plurality of the blockchain nodes at which a respective dataset is stored.

[0008] In some embodiments, these general and specific aspects may be implemented using a system, a method, or a computer program, or any combination of systems, methods, and computer programs. The foregoing and other described embodiments can each, optionally, include one or more of the following features:

[0009] A first feature, combinable with any of the following features, specifies that the index provides a correspondence between a dataset identifier (ID) of a dataset and a node ID of a blockchain node at which the dataset is stored.

[0010] A second feature, combinable with any of the previous or following features, specifies that the index provides a plurality of block IDs corresponding to the plurality of blocks that the data storage scheme is associated with.

[0011] A third feature, combinable with any of the previous or following features, further comprises: hashing a remainder of the plurality of datasets other than the at least one of the

plurality of datasets to generate hash values corresponding to the remainder of the plurality of datasets; storing the hash values; and deleting the one or more blocks. .

**[0012]** A fourth feature, combinable with any of the previous or following features, further comprises: receiving a request for blockchain data from a computing device; determining that the blockchain data is included in the one or more blocks; and sending, based on the index, hash values to a remainder of the blockchain nodes of the blockchain network to retrieve the remainder of the plurality of datasets..

**[0013]** A fifth feature, combinable with any of the previous or following features, further comprises: receiving at least one dataset from each of the remainder of the blockchain nodes; hashing the at least one dataset to generate at least one hash value corresponding to each of the remainder of the blockchain nodes; and determining whether the at least one hash value is stored in the blockchain node.

**[0014]** A sixth feature, combinable with any of the previous or following features, further comprises: in response to determining that the at least one hash value is not stored in the blockchain node, determining a blockchain node that the at least one dataset corresponding to the at least one hash value is received from; and reporting the blockchain node as a faulty node.

**[0015]** A seventh feature, combinable with any of the previous or following features, further comprises: in response to determining that the at least one hash value corresponding to each of the remainder of the blockchain nodes is stored in the blockchain node, decoding the one or more blocks based on the at least one of the plurality of datasets stored in the blockchain node and the at least one dataset received from each of the remainder of the blockchain nodes.

**[0016]** An eighth feature, combinable with any of the previous or following features, specifies that the one or more blocks are historical blocks that have been created for a predetermined amount of time.

**[0017]** A ninth feature, combinable with any of the previous or following features, specifies that the ECC is performed when utilization rate of computational resource of the blockchain node is less than or equal to a predetermined value or usage of storage space of the blockchain node is greater than or equal to a predetermined percentage.

**[0018]** A tenth feature, combinable with any of the previous or following features, specifies that the ECC is erasure coding performed by adding redundant bits to the plurality of blocks.

**[0019]** An eleventh feature, combinable with any of the previous or following features, specifies that the plurality of blocks are infrequently accessed blocks that are appended to the blockchain for a predetermined amount of time.

**[0020]** It is appreciated that methods in accordance with this specification may include any combination of the aspects and features described herein. That is, methods in accordance with this specification are not limited to the combinations of aspects and features specifically described herein, but also include any combination of the aspects and features provided.

**[0021]** The details of one or more embodiments of this specification are set forth in the accompanying drawings and the description below. Other features and advantages of this specification will be apparent from the description and drawings, and from the claims.

## BRIEF DESCRIPTION OF THE DRAWINGS

**[0022]** FIG. 1 depicts an example of an environment that can be used to execute embodiments of this specification.

**[0023]** FIG. 2 depicts an example of an architecture in accordance with embodiments of this specification.

**[0024]** FIG. 3 depicts an example of a block data encoding and hashing process in accordance with embodiments of this specification.

**[0025]** FIG. 4 depicts an example of a data storage scheme in accordance with embodiments of this specification.

**[0026]** FIG. 5 depicts another example of a block data encoding and hashing process in accordance with embodiments of this specification.

**[0027]** FIG. 6 depicts an example of a process for indexing encoded datasets in accordance with embodiments of this specification.

**[0028]** FIG. 7 depicts an example of a process for data retrieving and recovering in accordance with embodiments of this specification.

**[0029]** FIG. 8 depicts an example of a process that can be executed in accordance with embodiments of this specification.

**[0030]** FIG. 9 depicts examples of modules of an apparatus in accordance with embodiments of this specification.

**[0031]** Like reference numbers and designations in the various drawings indicate like elements.

## DETAILED DESCRIPTION

**[0032]** This specification describes technologies for indexing blockchain data for storage. These technologies generally involve generating one or more encoded blocks by executing error correction coding (ECC) on one or more blocks of a blockchain, dividing each of the one or more encoded blocks into a plurality of datasets, and providing an index for the one or more blocks, the index indexing each of the plurality of datasets to a blockchain node at which a respective dataset is stored.

**[0033]** As described herein, blockchain networks can store different types of data such as state data, block data, and index data. Block data includes all transactions in the blockchain network, which can take a large amount of storage space as new blocks are constantly adding to the blockchain. It can be inefficient for the blockchain nodes to each store an entire copy of the block data, especially for data of infrequently accessed blocks (e.g., blocks added to the blockchain long time ago). Accordingly, embodiments of this specification provide that each blockchain node stores a portion of infrequently accessed blocks and retrieves the rest of the block data from other blockchain nodes when needed, to reduce storage consumption. However, if faulty nodes or unreliable nodes exist in the blockchain network, the retrieved data cannot be trusted and data loss may occur.

**[0034]** In some embodiments, the blockchain nodes can perform ECC, such as erasure coding, to encode the infrequently accessed blocks. The ECC encoded blocks can then be divided into a plurality of datasets. The plurality of datasets can be indexed and assigned to different blockchain nodes to store based on a data storage scheme. When data from an infrequently accessed block is needed by a blockchain node to execute a smart contract, the blockchain node can retrieve corresponding datasets from other blockchain nodes based on the index to form the ECC encoded block and recover the original block. By sharing ECC encoded blocks, even if unauthentic data exists or data loss occurs, the original block data can be recovered as long as the percentage of honest blockchain nodes is greater than or equal to the code rate of the ECC.

**[0035]** The techniques described in this specification produce several technical effects. For example, embodiments of the subject matter reduce the burden on storage resources of blockchain nodes, while maintaining computational efficiency and data equality of the blockchain nodes. Because some blocks are infrequently accessed (e.g., older blocks), storage resources of blockchain nodes are conserved by saving only a portion of the ECC encoded blocks (also referred to herein as encoded blocks) on each blockchain node and sharing the remainder of the data with other blockchain nodes.

**[0036]** In some embodiments, an ECC encoded block can be divided to a plurality of datasets. A blockchain node can store a selected portion of the plurality of datasets and hash values corresponding to the remainder of the datasets. The selection can be based on a data storage scheme agreed to by blockchain nodes of the blockchain network. The plurality of datasets and the hash values can be indexed to block IDs associated with the ECC encoded blocks and node IDs associated with the blockchain nodes in which the respective datasets and hash values are stored. The index can be shared to the blockchain nodes of the blockchain network. When original blockchain data needs to be recovered by a blockchain node, the blockchain node can refer to the index to quickly locate where the datasets are stored, retrieve the datasets, and decode the original block that contains the blockchain data based on the datasets.

**[0037]** To retrieve the datasets, the blockchain node can send hash values of the datasets to other blockchain nodes that store the corresponding datasets. Since hash values are irreversible, the blockchain node can verify whether the retrieved datasets are authentic, by hashing the received datasets and comparing the hashed values with hash values that were locally stored. As such, data security can be ensured and faulty nodes can be identified.

**[0038]** To provide further context for embodiments of this specification, and as introduced above, distributed ledger systems (DLSs), which can also be referred to as consensus networks (e.g., made up of peer-to-peer nodes), and blockchain networks, enable participating entities to securely, and immutably conduct transactions, and store data. Although the term blockchain is generally associated with particular networks, and/or use cases, blockchain is used herein to generally refer to a DLS without reference to any particular use case.

**[0039]** A blockchain is a data structure that stores transactions in a way that the transactions are immutable. Thus, transactions recorded on a blockchain are reliable and trustworthy. A blockchain includes one or more blocks. Each block in the chain is linked to a previous block immediately before it in the chain by including a cryptographic hash of the previous block. Each block also includes a timestamp, its own cryptographic hash, and one or more transactions. The transactions, which have already been verified by the nodes of the blockchain network, are hashed and encoded into a Merkle tree. A Merkle tree is a data structure in which data at the leaf nodes of the tree is hashed, and all hashes in each branch of the tree are concatenated at the root of the branch. This process continues up the tree to the root of the entire tree, which stores a hash that is representative of all data in the tree. A hash purporting to be of a transaction stored in the tree can be quickly verified by determining whether it is consistent with the structure of the tree.

**[0040]** Whereas a blockchain is a decentralized or at least partially decentralized data structure for storing transactions, a blockchain network is a network of computing nodes that manage, update, and maintain one or more blockchains by broadcasting, verifying and validating transactions, etc. As introduced above, a blockchain network can be provided as a public blockchain network, a private blockchain network, or a consortium blockchain network. Embodiments of this specification are described in further detail herein with reference to a consortium blockchain network. It is contemplated, however, that embodiments of this specification can be realized in any appropriate type of blockchain network.

**[0041]** In general, a consortium blockchain network is private among the participating entities. In a consortium blockchain network, the consensus process is controlled by an authorized set of nodes, which can be referred to as consensus nodes, one or more consensus nodes being operated by a respective entity (e.g., a financial institution, insurance company). For example, a consortium of ten (10) entities (e.g., financial institutions, insurance companies) can operate a consortium blockchain network, each of which operates at least one node in the consortium blockchain network.

**[0042]** In some examples, within a consortium blockchain network, a global blockchain is provided as a blockchain that is replicated across all nodes. That is, all consensus nodes are in perfect state consensus with respect to the global blockchain. To achieve consensus (e.g.,

agreement to the addition of a block to a blockchain), a consensus protocol is implemented within the consortium blockchain network. For example, the consortium blockchain network can implement a practical Byzantine fault tolerance (PBFT) consensus, described in further detail below.

**[0043]** FIG. 1 is a diagram illustrating an example of an environment 100 that can be used to execute embodiments of this specification. In some examples, the environment 100 enables entities to participate in a consortium blockchain network 102. The environment 100 includes computing systems106, 108, and a network 110. In some examples, the network 110 includes a local area network (LAN), wide area network (WAN), the Internet, or a combination thereof, and connects web sites, user devices (e.g., computing devices), and back-end systems. In some examples, the network 110 can be accessed over a wired and/or a wireless communications link. In some examples, the network 110 enables communication with, and within the consortium blockchain network 102. In general the network 110 represents one or more communication networks. In some cases, the computing systems106, 108 can be nodes of a cloud computing system (not shown), or each of the computing systems106, 108 can be a separate cloud computing system including a number of computers interconnected by a network and functioning as a distributed processing system.

**[0044]** In the depicted example, the computing systems 106, 108 can each include any appropriate computing device that enables participation as a node in the consortium blockchain network 102. Examples of computing devices include, without limitation, a server, a desktop computer, a laptop computer, a tablet computing device, and a smartphone. In some examples, the computing systems 106, 108 host one or more computer-implemented services for interacting with the consortium blockchain network 102. For example, the computing system 106 can host computer-implemented services of a first entity (e.g., user A), such as a transaction management system that the first entity uses to manage its transactions with one or more other entities (e.g., other users). The computing system 108 can host computer-implemented services of a second entity (e.g., user B), such as a transaction management system that the second entity uses to manage its transactions with one or more other entities (e.g., other users). In the example of FIG. 1, the consortium blockchain network 102 is represented as a peer-to-peer network of nodes, and the computing systems 106, 108

provide nodes of the first entity, and second entity respectively, which participate in the consortium blockchain network 102.

**[0045]** FIG. 2 depicts an example of an architecture 200 in accordance with embodiments of this specification. The example conceptual architecture 200 includes participant systems 202, 204, 206 that correspond to Participant A, Participant B, and Participant C, respectively. Each participant (e.g., user, enterprise) participates in a blockchain network 212 provided as a peer-to-peer network including a plurality of nodes 214, at least some of which immutably record information in a blockchain 216. Although a single blockchain 216 is schematically depicted within the blockchain network 212, multiple copies of the blockchain 216 are provided, and are maintained across the blockchain network 212, as described in further detail herein.

**[0046]** In the depicted example, each participant system 202, 204, 206 is provided by, or on behalf of Participant A, Participant B, and Participant C, respectively, and functions as a respective node 214 within the blockchain network. As used herein, a node generally refers to an individual system (e.g., computer, server) that is connected to the blockchain network 212, and enables a respective participant to participate in the blockchain network. In the example of FIG. 2, a participant corresponds to each node 214. It is contemplated, however, that a participant can operate multiple nodes 214 within the blockchain network 212, and/or multiple participants can share a node 214. In some examples, the participant systems 202, 204, 206 communicate with, or through the blockchain network 212 using a protocol (e.g., hypertext transfer protocol secure (HTTPS)), and/or using remote procedure calls (RPCs).

**[0047]** Nodes 214 can have varying degrees of participation within the blockchain network 212. For example, some nodes 214 can participate in the consensus process (e.g., as miner nodes that add blocks to the blockchain 216), while other nodes 214 do not participate in the consensus process. As another example, some nodes 214 store a complete copy of the blockchain 216, while other nodes 214 only store copies of portions of the blockchain 216. For example, data access privileges can limit the blockchain data that a respective participant stores within its respective system. In the example of FIG. 2, the participant systems 202, 204, and 206 store respective, complete copies 216', 216'', and 216''' of the blockchain 216.

**[0048]** A blockchain (e.g., the blockchain 216 of FIG. 2) is made up of a chain of blocks, each block storing data. Examples of data include transaction data representative of a

transaction between two or more participants. While transactions are used herein by way of non-limiting example, it is contemplated that any appropriate data can be stored in a blockchain (e.g., documents, images, videos, audio). Examples of a transaction can include, without limitation, exchanges of something of value (e.g., assets, products, services, currency). The transaction data is immutably stored within the blockchain. That is, the transaction data cannot be changed.

**[0049]** Before storing in a block, the transaction data is hashed. Hashing is a process of transforming the transaction data (provided as string data) into a fixed-length hash value (also provided as string data). It is not possible to un-hash the hash value to obtain the transaction data. Hashing ensures that even a slight change in the transaction data results in a completely different hash value. Further, and as noted above, the hash value is of fixed length. That is, no matter the size of the transaction data the length of the hash value is fixed. Hashing includes processing the transaction data through a hash function to generate the hash value. An example of a hash function includes, without limitation, the secure hash algorithm (SHA)-256, which outputs 256-bit hash values.

**[0050]** Transaction data of multiple transactions are hashed and stored in a block. For example, hash values of two transactions are provided, and are themselves hashed to provide another hash. This process is repeated until, for all transactions to be stored in a block, a single hash value is provided. This hash value is referred to as a Merkle root hash, and is stored in a header of the block. A change in any of the transactions will result in change in its hash value, and ultimately, a change in the Merkle root hash.

**[0051]** Blocks are added to the blockchain through a consensus protocol. Multiple nodes within the blockchain network participate in the consensus protocol, and perform work to have a block added to the blockchain. Such nodes are referred to as consensus nodes. PBFT, introduced above, is used as a non-limiting example of a consensus protocol. The consensus nodes execute the consensus protocol to add transactions to the blockchain, and update the overall state of the blockchain network.

**[0052]** In further detail, the consensus node generates a block header, hashes all of the transactions in the block, and combines the hash value in pairs to generate further hash values until a single hash value is provided for all transactions in the block (the Merkle root hash). This hash is added to the block header. The consensus node also determines the hash value of

the most recent block in the blockchain (i.e., the last block added to the blockchain). The consensus node also adds a nonce value, and a timestamp to the block header.

**[0053]** In general, PBFT provides a practical Byzantine state machine replication that tolerates Byzantine faults (e.g., malfunctioning nodes, malicious nodes). This is achieved in PBFT by assuming that faults will occur (e.g., assuming the existence of independent node failures, and/or manipulated messages sent by consensus nodes). In PBFT, the consensus nodes are provided in a sequence that includes a primary consensus node, and backup consensus nodes. The primary consensus node is periodically changed. Transactions are added to the blockchain by all consensus nodes within the blockchain network reaching an agreement as to the world state of the blockchain network. In this process, messages are transmitted between consensus nodes, and each consensus nodes proves that a message is received from a specified peer node, and verifies that the message was not modified during transmission.

**[0054]** In PBFT, the consensus protocol is provided in multiple phases with all consensus nodes beginning in the same state. To begin, a client sends a request to the primary consensus node to invoke a service operation (e.g., execute a transaction within the blockchain network). In response to receiving the request, the primary consensus node multicasts the request to the backup consensus nodes. The backup consensus nodes execute the request, and each sends a reply to the client. The client waits until a threshold number of replies are received. In some examples, the client waits for f+1 replies to be received, where f is the maximum number of faulty consensus nodes that can be tolerated within the blockchain network. The final result is that a sufficient number of consensus nodes come to an agreement on the order of the record that is to be added to the blockchain, and the record is either accepted, or rejected.

**[0055]** In some blockchain networks, cryptography is implemented to maintain privacy of transactions. For example, if two nodes want to keep a transaction private, such that other nodes in the blockchain network cannot discern details of the transaction, the nodes can encrypt the transaction data. An example of cryptography includes, without limitation, symmetric encryption, and asymmetric encryption. Symmetric encryption refers to an encryption process that uses a single key for both encryption (generating ciphertext from plaintext), and decryption (generating plaintext from ciphertext). In symmetric encryption, the same key is available to multiple nodes, so each node can en-/de-crypt transaction data.

**[0056]** Asymmetric encryption uses keys pairs that each include a private key, and a public key, the private key being known only to a respective node, and the public key being known to any or all other nodes in the blockchain network. A node can use the public key of another node to encrypt data, and the encrypted data can be decrypted using other node's private key. For example, and referring again to FIG. 2, Participant A can use Participant B's public key to encrypt data, and send the encrypted data to Participant B. Participant B can use its private key to decrypt the encrypted data (ciphertext) and extract the original data (plaintext). Messages encrypted with a node's public key can only be decrypted using the node's private key.

**[0057]** Asymmetric encryption is used to provide digital signatures, which enables participants in a transaction to confirm other participants in the transaction, as well as the validity of the transaction. For example, a node can digitally sign a message, and another node can confirm that the message was sent by the node based on the digital signature of Participant A. Digital signatures can also be used to ensure that messages are not tampered with in transit. For example, and again referencing FIG. 2, Participant A is to send a message to Participant B. Participant A generates a hash of the message, and then, using its private key, encrypts the hash to provide a digital signature as the encrypted hash. Participant A appends the digital signature to the message, and sends the message with digital signature to Participant B. Participant B decrypts the digital signature using the public key of Participant A, and extracts the hash. Participant B hashes the message and compares the hashes. If the hashes are same, Participant B can confirm that the message was indeed from Participant A, and was not tampered with.

**[0058]** FIG. 3 depicts an example of a block data encoding and hashing process 300 in accordance with embodiments of this specification. In this example, a blockchain network of four blockchain nodes is depicted, which are blockchain nodes 302, 304, 306, and 308. Using blockchain node 302 as an example to illustrate the encoding and hashing process 300, the blockchain node 302 can store block data of the blockchain network to block body of a block 312. In the illustrated example, the block data is stored in block 100. Afterwards, the blockchain node 302 can engage in a consensus process with other blockchain nodes 304, 306, and 308. During the consensus process, the blockchain node 302 can perform a

consensus algorithm, such as proof of work (PoW) or proof of stake (PoS) to create a corresponding block on the blockchain.

**[0059]** In some embodiments, the blockchain node 302 can identify one or more infrequently accessed blocks. In practice, the longer a block has been created, the less likely the corresponding block data is needed for operations such as executing smart contracts. The blockchain node 302 can determine that locally stored blocks are infrequently accessed when they are historical blocks that have been created on the blockchain for a predetermined amount of time. For example, the predetermined amount of time can be one or two times of the average time a block is created. In some examples, a block can also be determined as infrequently accessed when no block data in the block is retrieved for the predetermined amount of time to execute smart contracts.

**[0060]** After identifying infrequently accessed blocks, the blockchain node 302 can perform ECC 314 of block data in the block body of each of the infrequently accessed blocks. ECC can be used for controlling errors or losses of data over unreliable transmissions by adding redundant bits (also referred to as redundancy) to the data. Redundant bits can be a complex function of many original information bits. The redundancy can allow errors or losses of data to be corrected without retransmission of the data. The original information may or may not appear literally in the encoded output. ECC codes that include the unmodified original information in the encoded output are referred to as systematic ECC codes, while those that do not are referred to as non-systematic ECC codes. The maximum fractions of errors or of missing bits that can be corrected by ECC is determined by the design of the ECC code. Therefore, different error correction codes are suitable for different conditions. In general, a stronger ECC code induces more redundancy, which increases storage consumption of the code and reduces communication efficiency if the encoded information is to be transmitted.

**[0061]** One example ECC can be the erasure coding. Using the erasure coding, a message of k symbols can be encoded to a codeword with n symbols, where k and n are natural numbers, and $k < n$. The message can be recovered from a subset of the n-symbol codeword. The fraction $r = k/n$ is the code rate of the erasure code.

**[0062]** By using ECC, each of the blockchain nodes can store a portion of the encoded block data and retrieve the rest of the encoded block data from other blockchain nodes when

needed. In some embodiments, the ECC can be performed when utilization rate of computational resource of the blockchain node 302 is lower than a predetermined value (e.g., 40%). As such, the interference with other computational operations on the blockchain node 302 can be reduced. In some embodiments, ECC can be performed when the usage of storage space of the blockchain node 302 is greater than or equal to a predetermined percentage, such that after ECC, some portions of the encoded block data can be deleted to free up storage space.

**[0063]** Again, using block 100 as an example, assuming that the blockchain node 302 determines the block 100 as an infrequently accessed block and performs ECC 314, the ECC encoded data can be divided into a plurality of datasets based on a data storage scheme. A data storage scheme can be provided as a set of computer-executable instructions that define where and/or how data is to be stored within the blockchain network. In some examples, the data storage scheme can be provided by a trusted node with proof of authority and agreed to by the blockchain nodes. In some examples, the data storage scheme can be agreed to by the blockchain nodes through consensus. Generally, the data storage scheme can include one or more predetermined rules for dividing the encoded data to a plurality of datasets based on the number of blockchain nodes in a blockchain network. The data storage scheme can also include assignments of one or more datasets of the plurality of datasets to be stored or hashed by each of the blockchain nodes. To ensure data equality, the data storage scheme can include an assignment of at least one dataset to be stored by each blockchain node of the blockchain network.

**[0064]** In the example shown in FIG. 3, the encoded block data of block 100 is divided into four datasets, which are Data1, Data2, Data3, and Vdata1, each to be stored by one of the blockchain nodes 302, 304, 306, and 308. Vdata1 can represent the redundant bits of the ECC for error correction. Data1 is selected to be stored by the blockchain node 302 according to the data storage scheme. Data2, Data3, and Vdata1 are selected to be separately hashed 316 to generate hash values Dhash2, Dhash3, and Vhash1, respectively. In accordance with embodiments of this specification, the encoded data can be divided to more than four datasets when the blockchain network has more than four nodes. In some examples, each of the blockchain nodes can store more than one dataset and hash the rest of the datasets assigned to be stored by other nodes.

**[0065]** Referring now to FIG. 4, FIG. 4 depicts an example of a data storage scheme 400 in accordance with embodiments of this specification. As discussed earlier, Data1 is selected to be stored by the blockchain node 302 according to the data storage scheme 400. Based on the data storage scheme 400, blockchain node 304 stores Data2 and separately hashes Data1, Data3, and Vdata1 to generate hash values Dhash1, Dhash3, and Vhash1, respectively. Blockchain node 306 stores Data3 and separately hashes Data1, Data2, and Vdata1 to generate hash values Dhash1, Dhash2 and Vhash1, respectively. Blockchain node 308 stores Vdata1 and separately hashes Data1, Data2, and Vdata3 to generate hash values Dhash1, Dhash2 and Dhash3, respectively.

**[0066]** Referring back to FIG. 3, because the hash values correspond to encoded datasets of the same block, they can be indexed by a block ID of the block. For example, the blockchain node 302 can index Data1, Dhash1, Dhash2, and Vhash1 associated with block 100 with a block ID 100. As such, the blockchain node 302 can use the indexed block ID to map the hash values to their corresponding blocks. A more detailed example of indexing the datasets and hash values is discussed in the description of FIG. 6.

**[0067]** It is to be understood that other data storage schemes can be made for the blockchain nodes 302, 304, 306, and 308, according to the data storage scheme. In some examples, the encoded block data of block 100 can be divided to more than four datasets. It is to be understood that other data storage schemes can be made for the blockchain nodes 502, 504, 506, and 508, according to the data storage scheme.

**[0068]** After generating and storing Dhash2, Dhash3, and Vhash1, the blockchain node 302 can delete Data2, Data3, and Vdata1 from storage to save storage space. As such, for each block, the blockchain node 302 only stores one ECC encoded dataset (i.e., Data1) and three hash values (i.e., Dhash2, Dhash3, and Vhash1), instead of the entire block. As such, storage space can be significantly reduced. Similar to block 100, the encoding and hashing process can be performed for other infrequently accessed blocks that are stored by the blockchain nodes 304, 306, and 308.

**[0069]** When the blockchain node 302 determines that block data of the block 100 is needed for executing a smart contract, it can retrieve Data2, Data3, and Vdata1 from blockchain nodes 304, 306, and 308, respectively, according to the data storage scheme. To retrieve datasets from other blockchain nodes 304, 306, and 308, blockchain node 302 can

send hash values corresponding to the datasets to be retrieved according to the data storage scheme.

**[0070]** For example, to retrieve Data2, the blockchain node 302 can send Dhash2 to the blockchain node 304. If the blockchain node 304 has Data2 stored, it can send the Data2 back to the blockchain node 302 in response to receiving the Dhash2. After receiving the Data2 from the blockchain node 304, the blockchain node 302 can hash the received dataset and compare the hash value with Dhash2. If the hash value is the same as Dhash2, the blockchain node 302 can determine that the received dataset is authentic. Otherwise, the received dataset is determined to be unauthentic. When the received dataset is determined as unauthentic, the blockchain node 302 can report the blockchain node 304 as a faulty node (or a Byzantine node). If the percentage of unauthentic data received by the blockchain node 302 is less than or equal to the maximum fraction of erroneous or missing bits that can be corrected by the ECC, block 100 can be recovered from the locally stored and received datasets.

**[0071]** As described earlier, blockchain networks can store different types of data such as state data, block data, and index data. State data are often stored as a content-addressed state tree, such as the MPT or the fixed depth Merkle tree (FDMT). Content-addressed state trees are incremental in nature. That is, changes of account states are reflected by adding new tree structures instead of only updating values of the existing state tree. Therefore, the content-addressed state trees can grow very large in size when blocks are continuously added to the blockchain. Under the FDMT storage scheme, state data can be separated into current state data associated with the current block and historic state data associated with all blocks of the blockchain. Most data in the FDMT are infrequently used historic state data. Storing all historic state data in every consensus node can be quite inefficient in terms of storage resource usage.

**[0072]** In some embodiments, similar to encoding and sharing block data, ECC such as erasure coding can be used to encode the historic state data. Each consensus node in the blockchain network stores only a portion of the historic state data and retrieves the rest of the historic state data from other nodes to reduce storage consumption. By sharing ECC encoded historic state data instead of the original historic state data, even if unauthentic data exists or data loss occurs, the original historic state data can be recovered, as long as the percentage of

unauthentic data or data loss is less than or equal to the maximum fraction of erroneous or missing bits that can be corrected by the ECC.

**[0073]** FIG. 5 depicts another example of a block data encoding and hashing process 500 in accordance with embodiments of this specification. In this example, a blockchain network of four blockchain nodes is depicted, which are blockchain nodes 502, 504, 506, and 508. Using blockchain node 502 as an example to illustrate the encoding and hashing process 500, when new block data are added to the block 512, the blockchain node 502 can perform ECC 514 to encode the block data. As compared to the encoding and hashing process 300 discussed in the description of FIG. 3, the blockchain node 502 performs ECC on the block data as they are written to a block. As such, the blockchain node 502 does not need to store the entire block, but can instead, store a selected portion of the ECC encoded block data and hash values corresponding to the rest of the encoded block data based on the data storage scheme. This encoding and hashing process 500 can be especially suitable for scenarios when blockchain node 502 has low disk space.

**[0074]** In some embodiments, instead of storing data as blocks, the blockchain node 502 can store a write-ahead log (WAL) file or other similar roll-forward journal files. The WAL file can record block data that have been committed but not yet stored by the blockchain node 502. Using the WAL file, the original blockchain data can be preserved in the database file, while changes of the blockchain data can be written into a separate WAL file. A commit to roll-forward with the changes can happen without ever writing to the original blockchain data. This arrangement allows continued operations of the blockchain data while changes are committed into the WAL file. By using the WAL file to store changes made through the encoding and hashing process 500, the blockchain node 502 can indicate that it has the block data for consensus, while performing the ECC in the background when appropriate. As such, the ECC can be performed when utilization rate of computational resource of the blockchain node 302 is low, in order to reduce the impact on computational efficiency or latency of the consensus process.

**[0075]** In some embodiments, the blockchain node 502 can store the block data in a buffer. The blockchain node 502 can perform ECC to the block data stored in the buffer when the size of the data is greater than a predetermined threshold or when the buffer is full. After performing ECC, the blockchain node 502 can follow the encoding and hashing

process 500 to store encoded block data and hash values, as discussed in the description below.

**[0076]** Using block 100 as an example again, after performing the ECC, the encoded block data can be divided into a plurality of datasets based on the data storage scheme. Similar to the example discussed in the description of FIG. 3, the encoded block data of block 100 can be divided into four datasets, which are Data1, Data2, Data3, and Vdata1, each to be stored by one of the blockchain nodes 502, 504, 506, and 508. Vdata1 can represent the redundant bits of the ECC. Data1 is selected to be stored by the blockchain node 502 according to the data storage scheme. Data2, Data3, and Vdata1 are selected to be separately hashed 516 to generate hash values Dhash2, Dhash3, and Vhash1, respectively.

**[0077]** The hash values can be indexed by a block ID of a corresponding block of the hash values. For example, the blockchain node 502 can index Data1, Dhash1, Dhash2, and Vhash1 associated with block 100 with a block ID 100. As such, the blockchain node 502 can use the indexed block ID to map the hash values to their corresponding blocks. A more detailed example of indexing the datasets and hash values is discussed in the description of FIG. 6.

**[0078]** It is to be understood that other data storage schemes can be made for the one or more blockchain nodes 502, 504, 506, and 508, according to the data storage scheme. For example, the encoded block data of block 100 can be divided into more than four datasets. Each of the blockchain nodes 502, 504, 506, and 508 can store more than one dataset and hash the rest of the datasets stored by other nodes.

**[0079]** After generating Dhash2, Dhash3, and Vhash1, the blockchain node 502 can store Data1, Dhash2, Dhash3, and Vhash1 and delete Data2, Data3, and Vdata1 from storage to save storage space. As such, for each block of the blockchain, the blockchain node 502 only stores one dataset (i.e., Data1) and three hash values (i.e., Dhash2, Dhash3, and Vhash1) of the ECC encoded block data instead of the original block data to save on storage space. When the blockchain node 502 determines that block data of the block 100 is needed for executing a smart contract, it can retrieve Data2, Data3, and Vdata1 from blockchain nodes 504, 506, and 508, respectively, according to the data storage scheme.

**[0080]** To retrieve datasets from other blockchain nodes 504, 506, and 508, blockchain node 502 can send hash values corresponding to the datasets to be retrieved according to the

data storage scheme. For example, to retrieve Data2, the blockchain node 502 can send Dhash2 to the blockchain node 504. If the blockchain node 504 has Data2 stored, it can send the Data2 back to the blockchain node 502 in response to receiving the Dhash2. After receiving the Data2 from the blockchain node 504, the blockchain node 502 can hash the received dataset and compare the hash value with Dhash2. If the hash value is the same as Dhash2, the blockchain node 502 can determine that the received dataset is authentic. Otherwise, the received dataset can be determined as unauthentic. When the received dataset is determined as unauthentic, the blockchain node 502 can report the blockchain node 504 as a faulty node (or a Byzantine node). If the percentage of unauthentic data received by the blockchain node 502 is less than or equal to the maximum fraction of erroneous or missing bits that can be corrected by the ECC, block 100 can be recovered from the locally stored and received datasets.

**[0081]** As discussed earlier, by performing the encoding and hashing process, blockchain data can be ECC encoded and divided into a plurality of datasets. To save on storage space, each blockchain node can store one or more of the plurality of datasets and hash values of rest of the datasets based on a data storage scheme. The stored datasets and hash values can be indexed with Block IDs in order for a blockchain node to retrieve datasets from other nodes to recover original data.

**[0082]** FIG. 6 depicts an example of a process 600 for indexing encoded datasets in accordance with embodiments of this specification. The process 600 can be performed by a blockchain node of a blockchain network. The blockchain network can include four blockchain nodes, denoted as Node1, Node2, Node3, and Node4. In the depicted example, blocks 91-100 are used to illustrate the process 699 for indexing ECC encoded datasets. The blocks 91-100 can be infrequently accessed blocks as discussed in the description of FIG. 3. In some cases, the blocks 91-100 can also represent historical state data under the FDMT structure corresponding to blocks 91-100. In such cases, the process 600 can be performed to index datasets divided from encoded historical state data.

**[0083]** In some embodiments, a blockchain node can perform ECC 604 to each of the blocks 91-100 to generate ECC encoded blocks 602. The blockchain node can then divide each of the ECC encoded blocks 602 into four datasets 608, which are Data1, Data2, Data3, and Vdata1, where Vdata1 corresponds to the redundant bits for error correction. The

division of each of the ECC encoded blocks 602 can be performed based on a data storage scheme as discussed in the descriptions of FIGS. 3 and 4. The data storage scheme can provide how the ECC encoded blocks 602 should be divided into datasets. The data storage scheme can also provide assignments of the datasets to the blockchain nodes.

**[0084]** After the datasets 608 are generated for each of the blocks 91-100, the blockchain node can perform indexing 606 of the datasets 608 by creating an index table 610. The index table 610 can be created based on the data storage scheme agreed to by the blockchain nodes of the blockchain network.

**[0085]** The index table can include block IDs of blocks that are subject to the data storage scheme and the dataset(s) to be stored by each of the blockchain nodes. For example, the index table 610 can indicate data storage scheme of blocks 91-100. The ECC encoded data is divided to four datasets including Data1, Data2, Data3, and Vdata1. For each of blocks 91-100, Node1 stores Data1, Node2 stores Data2, Node3 stores Data3, and Node4 stores Vdata1. Therefore, instead of storing the entire blocks, each blockchain node stores only one dataset of each encoded block according to the index table 610.

**[0086]** FIG. 7 depicts an example of a process 700 for data retrieving and recovering in accordance with embodiments of this specification. When a client device 710 needs to get block or state data to execute a smart contract, it can send a request to a blockchain node (e.g., Node1 720a) to get the block data or state data from the blockchain. Upon receiving the request, Node1 can identify one or more blocks that include the requested data. Assuming that the requested data is included in block 100, the Node1 720a can read index table 714 that indicates data storage schemes for blocks 91-100. According to the index table 714, Node1 stores Data1 and Vhash1. Node2 720b stores Data2, Node3 720c stores Data3, and Node4 720d stores Vdata1. To collect the entire ECC encoded block 100, Node1 can then get Data2 from Node2 720b, Data3 from Node3 720c, and Vdata1 from Node4 720d. To get Data2, Data3, and Vdata1, Node1 720a can send requests that includes hash values of the datasets to the corresponding blockchain nodes. For example, by receiving the hash value of Data2 from Node1 720a, Node2 720b can determine that Data2 is requested and return it to Node1 720a.

**[0087]** After receiving the datasets from Node2 720b, Node3 720c, and Node4 720d, Node1 720a can verify the received datasets by generating hash values of the received datasets and compare them with hash values stored in a hash table 716. The hash table 716

indicates a mapping relationship between the datasets and their hash values, namely, Data2 and its hash value Dhash2, Data3 and its hash value Dhash3, and Vdata1 and its hash value Vhash1. If the hash value of the dataset received from Node2 720b matches Dhash2, the hash value of the dataset received from Node3 720c matches Dhash3, and the hash value of the dataset received from Node4 matches Vhash1, the Node1 720a can determine that the received data are authentic. The Node1 720a can combine the received data (i.e., Data2, Data3, and Vdata1) with the Data1 it stores to form the ECC encoded block 100. The Node1 720 can then decode the ECC encoded block to recover block 100. If any of the hash values does not match, the Node1 720a can report that the corresponding blockchain node that sends the dataset is a faulty node. After block 100 is recovered, the Node1 720a can return the requested block or state data in block 100 to the client device 710.

**[0088]** FIG. 8 is a flowchart of an example of a process 800 for communicating and sharing blockchain data. For convenience, the process 800 will be described as being performed by a blockchain node. The blockchain node can be a computer or a system of one or more computers, located in one or more locations, and programmed appropriately in accordance with this specification. For example, a computing device in a computing system, e.g., the computing system 106, 108 of FIG. 1, appropriately programmed, can be a blockchain node that performs the process 800.

**[0089]** At 802, a blockchain node generates a plurality of encoded blocks based on performing ECC on a plurality of blocks of a blockchain. In some examples, the one or more blocks are infrequent accessed blocks that have been appended to the blockchain for a predetermined amount of time. In some examples, the ECC is performed when utilization rate of computational resource of the blockchain node is less than or equal to a predetermined value or usage of storage space of the blockchain node is greater than or equal to a predetermined percentage. In some examples, the ECC is performed by adding redundant bits to the one or more blocks. In some examples, the ECC is erasure coding.

**[0090]** At 804, for each encoded block of the plurality of encoded blocks, the blockchain node divides the encoded block into a plurality of datasets based on a data storage scheme associated with the plurality of blocks, wherein the data storage scheme provides assignments of the plurality of datasets to a plurality of blockchain nodes.

**[0091]** At 806, the blockchain nodes stores at least one of the plurality of datasets based on the assignments provided in the data storage scheme.

**[0092]** At 808, the blockchain node provides an index that indexes each of the plurality of datasets to each of the plurality of the blockchain nodes at which a respective dataset is stored. In some examples, the index provides a correspondence between a dataset ID of a dataset and a node ID of a blockchain node at which the dataset is stored. In some examples, the index provides a plurality of block IDs corresponding to the plurality of blocks that the data storage scheme is associated with.

**[0093]** In some examples, the process 800 further comprises: hashing a remainder of the plurality of datasets other than the at least one of the plurality of datasets to generate hash values corresponding to the remainder of the plurality of datasets; storing the hash values; and deleting the one or more blocks.

**[0094]** In some examples, the process 800 further comprises: receiving a request for blockchain data from a computing device; determining that the blockchain data is included in the one or more blocks; and sending, based on the index, hash values to a remainder of the blockchain nodes of the blockchain network to retrieve the remainder of the plurality of datasets.

**[0095]** In some examples, the process 800 further comprises: receiving at least one dataset from each of the remainder of the blockchain nodes; hashing the at least one dataset to generate at least one hash value corresponding to each of the remainder of the blockchain nodes; and determining whether the at least one hash value is stored in the blockchain node.

**[0096]** In some examples, the process 800 further comprising: in response to determining that the at least one hash value is not stored in the blockchain node, determining a blockchain node that the at least one dataset corresponding to the at least one hash value is received from; and reporting the blockchain node as a faulty node.

**[0097]** In some examples, the process 800 further comprising: in response to determining that the at least one hash value corresponding to each of the remainder of the blockchain nodes is stored in the blockchain node, decoding the one or more blocks based on the at least one of the plurality of datasets stored in the blockchain node and the at least one dataset received from each of the remainder of the blockchain nodes.

**[0098]** FIG. 9 is a diagram of an example of modules of an apparatus 900 in accordance with embodiments of this specification.

**[0099]** The apparatus 900 can be an example of an embodiment of a blockchain node configured to communicate and share blockchain data. The apparatus 900 can correspond to the embodiments described above, and the apparatus 900 includes the following: a generating module 902 that generates one or more encoded blocks by executing ECC on one or more blocks of a blockchain; a dividing module 904 that divides the encoded block into a plurality of datasets based on a data storage scheme associated with the plurality of blocks, wherein the data storage scheme provides assignments of the plurality of datasets to a plurality of blockchain nodes; a storing module 906 that stores at least one of the plurality of datasets based on the assignments provided in the data storage scheme; and an indexing module 908 that provides an index for the one or more blocks, the index indexing each of the plurality of datasets to a blockchain node at which a respective dataset is stored.

**[0100]** In an optional embodiment, the index provides a correspondence between a dataset ID of a dataset and a node ID of a blockchain node at which the dataset is stored.

**[0101]** In an optional embodiment, the index provides a plurality of block IDs corresponding to the plurality of blocks that the data storage scheme is associated with.

**[0102]** In an optional embodiment, the apparatus 900 further comprising: hashing a remainder of the plurality of datasets other than the at least one of the plurality of datasets to generate hash values corresponding to the remainder of the plurality of datasets; storing the hash values; and deleting the one or more blocks.

**[0103]** In an optional embodiment, the apparatus 900 further comprising: receiving a request for blockchain data from a computing device; determining that the blockchain data is included in the one or more blocks; and sending, based on the index, hash values to a remainder of the blockchain nodes of the blockchain network to retrieve the remainder of the plurality of datasets.

**[0104]** In an optional embodiment, the apparatus 900 further comprising: receiving at least one dataset from each of the remainder of the blockchain nodes; hashing the at least one dataset to generate at least one hash value corresponding to each of the remainder of the blockchain nodes; and determining whether the at least one hash value is stored in the blockchain node.

**[0105]** In an optional embodiment, the apparatus 900 further comprising: in response to determining that the at least one hash value is not stored in the blockchain node, determining a blockchain node that the at least one dataset corresponding to the at least one hash value is received from; and reporting the blockchain node as a faulty node.

**[0106]** In an optional embodiment, the apparatus 900 further comprising: in response to determining that the at least one hash value corresponding to each of the remainder of the blockchain nodes is stored in the blockchain node, decoding the one or more blocks based on the at least one of the plurality of datasets stored in the blockchain node and the at least one dataset received from each of the remainder of the blockchain nodes.

**[0107]** In an optional embodiment, the one or more blocks are historical blocks that have been created for a predetermined amount of time.

**[0108]** In an optional embodiment, the ECC is performed when utilization rate of computational resource of the blockchain node is less than or equal to a predetermined value or usage of storage space of the blockchain node is greater than or equal to a predetermined percentage.

**[0109]** In an optional embodiment, the ECC is erasure coding performed by adding redundant bits to the plurality of blocks.

**[0110]** In an optional embodiment, the plurality of blocks are infrequently accessed blocks that are appended to the blockchain for a predetermined amount of time.

**[0111]** The system, apparatus, module, or unit illustrated in the previous embodiments can be implemented by using a computer chip or an entity, or can be implemented by using a product having a certain function. A typical embodiment device is a computer, and the computer can be a personal computer, a laptop computer, a cellular phone, a camera phone, a smartphone, a personal digital assistant, a media player, a navigation device, an email receiving and sending device, a game console, a tablet computer, a wearable device, or any combination of these devices.

**[0112]** For an embodiment process of functions and roles of each module in the apparatus, references can be made to an embodiment process of corresponding steps in the previous method. Details are omitted here for simplicity.

**[0113]** Because an apparatus embodiment basically corresponds to a method embodiment, for related parts, references can be made to related descriptions in the method embodiment.

The previously described apparatus embodiment is merely an example. The modules described as separate parts may or may not be physically separate, and parts displayed as modules may or may not be physical modules, may be located in one position, or may be distributed on a number of network modules. Some or all of the modules can be selected based on actual demands to achieve the objectives of the solutions of the specification. A person of ordinary skill in the art can understand and implement the embodiments of the present application without creative efforts.

**[0114]** Referring again to FIG. 9, it can be interpreted as illustrating an internal functional module and a structure of a blockchain node. An execution body in essence can be an electronic device, and the electronic device includes the following: one or more processors; and one or more computer-readable memories configured to store an executable instruction of the one or more processors. In some embodiments, the one or more computer-readable memories are coupled to the one or more processors and have programming instructions stored thereon that are executable by the one or more processors to perform algorithms, methods, functions, processes, flows, and procedures as described in this specification. This specification also provides one or more non-transitory computer-readable storage media coupled to one or more processors and having instructions stored thereon which, when executed by the one or more processors, cause the one or more processors to perform operations in accordance with embodiments of the methods provided herein.

**[0115]** This specification further provides a system for implementing the methods provided herein. The system includes one or more processors, and a computer-readable storage medium coupled to the one or more processors having instructions stored thereon which, when executed by the one or more processors, cause the one or more processors to perform operations in accordance with embodiments of the methods provided herein.

**[0116]** Embodiments of the subject matter and the actions and operations described in this specification can be implemented in digital electronic circuitry, in tangibly-embodied computer software or firmware, in hardware, including the structures disclosed in this specification and their structural equivalents, or in combinations of one or more of them. Embodiments of the subject matter described in this specification can be implemented as one or more computer programs, e.g., one or more modules of computer program instructions, encoded on a computer program carrier, for execution by, or to control the operation of, data

processing apparatus. For example, a computer program carrier can include one or more computer-readable storage media that have instructions encoded or stored thereon. The carrier may be a tangible non-transitory computer-readable medium, such as a magnetic, magneto optical, or optical disk, a solid state drive, a random access memory (RAM), a read-only memory (ROM), or other types of media. Alternatively, or in addition, the carrier may be an artificially generated propagated signal, e.g., a machine-generated electrical, optical, or electromagnetic signal that is generated to encode information for transmission to suitable receiver apparatus for execution by a data processing apparatus. The computer storage medium can be or be part of a machine-readable storage device, a machine-readable storage substrate, a random or serial access memory device, or a combination of one or more of them. A computer storage medium is not a propagated signal.

**[0117]** A computer program, which may also be referred to or described as a program, software, a software application, an app, a module, a software module, an engine, a script, or code, can be written in any form of programming language, including compiled or interpreted languages, or declarative or procedural languages; and it can be deployed in any form, including as a stand-alone program or as a module, component, engine, subroutine, or other unit suitable for executing in a computing environment, which environment may include one or more computers interconnected by a data communication network in one or more locations.

**[0118]** A computer program may, but need not, correspond to a file in a file system. A computer program can be stored in a portion of a file that holds other programs or data, e.g., one or more scripts stored in a markup language document, in a single file dedicated to the program in question, or in multiple coordinated files, e.g., files that store one or more modules, sub programs, or portions of code.

**[0119]** Processors for execution of a computer program include, by way of example, both general- and special-purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive the instructions of the computer program for execution as well as data from a non-transitory computer-readable medium coupled to the processor.

**[0120]** The term "data processing apparatus" encompasses all kinds of apparatuses, devices, and machines for processing data, including by way of example a programmable

processor, a computer, or multiple processors or computers. Data processing apparatus can include special-purpose logic circuitry, e.g., an FPGA (field programmable gate array), an ASIC (application specific integrated circuit), or a GPU (graphics processing unit). The apparatus can also include, in addition to hardware, code that creates an execution environment for computer programs, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them.

**[0121]** The processes and logic flows described in this specification can be performed by one or more computers or processors executing one or more computer programs to perform operations by operating on input data and generating output. The processes and logic flows can also be performed by special-purpose logic circuitry, e.g., an FPGA, an ASIC, or a GPU, or by a combination of special-purpose logic circuitry and one or more programmed computers.

**[0122]** Computers suitable for the execution of a computer program can be based on general or special-purpose microprocessors or both, or any other kind of central processing unit. Generally, a central processing unit will receive instructions and data from a read only memory or a random access memory or both. Elements of a computer can include a central processing unit for executing instructions and one or more memory devices for storing instructions and data. The central processing unit and the memory can be supplemented by, or incorporated in, special-purpose logic circuitry.

**[0123]** Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to one or more storage devices. The storage devices can be, for example, magnetic, magneto optical, or optical disks, solid state drives, or any other type of non-transitory, computer-readable media. However, a computer need not have such devices. Thus, a computer may be coupled to one or more storage devices, such as, one or more memories, that are local and/or remote. For example, a computer can include one or more local memories that are integral components of the computer, or the computer can be coupled to one or more remote memories that are in a cloud network. Moreover, a computer can be embedded in another device, e.g., a mobile telephone, a personal digital assistant (PDA), a mobile audio or video player, a game console, a Global Positioning System (GPS) receiver, or a portable storage device, e.g., a universal serial bus (USB) flash drive, to name just a few.

**[0124]** Components can be "coupled to" each other by being commutatively such as electrically or optically connected to one another, either directly or via one or more intermediate components. Components can also be "coupled to" each other if one of the components is integrated into the other. For example, a storage component that is integrated into a processor (e.g., an L2 cache component) is "coupled to" the processor.

**[0125]** To provide for interaction with a user, embodiments of the subject matter described in this specification can be implemented on, or configured to communicate with, a computer having a display device, e.g., a LCD (liquid crystal display) monitor, for displaying information to the user, and an input device by which the user can provide input to the computer, e.g., a keyboard and a pointing device, e.g., a mouse, a trackball or touchpad. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback, e.g., visual feedback, auditory feedback, or tactile feedback; and input from the user can be received in any form, including acoustic, speech, or tactile input. In addition, a computer can interact with a user by sending documents to and receiving documents from a device that is used by the user; for example, by sending web pages to a web browser on a user's device in response to requests received from the web browser, or by interacting with an app running on a user device, e.g., a smartphone or electronic tablet. Also, a computer can interact with a user by sending text messages or other forms of message to a personal device, e.g., a smartphone that is running a messaging application, and receiving responsive messages from the user in return.

**[0126]** This specification uses the term "configured to" in connection with systems, apparatus, and computer program components. For a system of one or more computers to be configured to perform particular operations or actions means that the system has installed on it software, firmware, hardware, or a combination of them that in operation cause the system to perform the operations or actions. For one or more computer programs to be configured to perform particular operations or actions means that the one or more programs include instructions that, when executed by data processing apparatus, cause the apparatus to perform the operations or actions. For special-purpose logic circuitry to be configured to perform particular operations or actions means that the circuitry has electronic logic that performs the operations or actions.

**[0127]** While this specification contains many specific embodiment details, these should not be construed as limitations on the scope of what is being claimed, which is defined by the claims themselves, but rather as descriptions of features that may be specific to particular embodiments. Certain features that are described in this specification in the context of separate embodiments can also be realized in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiments can also be realized in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially be claimed as such, one or more features from a claimed combination can in some examples be excised from the combination, and the claim may be directed to a subcombination or variation of a subcombination.

**[0128]** Similarly, while operations are depicted in the drawings and recited in the claims in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. In certain circumstances, multitasking and parallel processing may be advantageous. Moreover, the separation of various system modules and components in the embodiments described above should not be understood as requiring such separation in all embodiments, and it should be understood that the described program components and systems can generally be integrated together in a single software product or packaged into multiple software products.

**[0129]** Particular embodiments of the subject matter have been described. Other embodiments are within the scope of the following claims. For example, the actions recited in the claims can be performed in a different order and still achieve desirable results. As one example, the processes depicted in the accompanying figures do not necessarily require the particular order shown, or sequential order, to achieve desirable results. In some examples, multitasking and parallel processing may be advantageous.

# CLAIMS

1. A computer-implemented method for indexing blockchain data for storage, the method comprising:

generating a plurality of encoded blocks based on performing error correction coding (ECC) on a plurality of blocks of a blockchain;

for each encoded block of the plurality of encoded blocks:

dividing the encoded block into a plurality of datasets based on a data storage scheme associated with the plurality of blocks, wherein the data storage scheme provides assignments of the plurality of datasets to a plurality of blockchain nodes;

storing at least one of the plurality of datasets based on the assignments provided in the data storage scheme; and

providing an index that indexes each of the plurality of datasets to each of the plurality of the blockchain nodes at which a respective dataset is stored.

2. The method of claim 1, wherein the index provides a correspondence between a dataset identifier (ID) of a dataset and a node ID of a blockchain node at which the dataset is stored.

3. The method of any previous claim, wherein the index provides a plurality of block IDs corresponding to the plurality of blocks that the data storage scheme is associated with.

4. The method of any previous claim, further comprising:

hashing a remainder of the plurality of datasets other than the at least one of the plurality of datasets to generate hash values corresponding to the remainder of the plurality of datasets;

storing the hash values; and

deleting the one or more blocks.

5. The method of claim 4, further comprising:

receiving a request for blockchain data from a computing device;

determining that the blockchain data is included in the one or more blocks; and

sending, based on the index, hash values to a remainder of the blockchain nodes of the blockchain network to retrieve the remainder of the plurality of datasets.

6. The method of claim 5, further comprising:

receiving at least one dataset from each of the remainder of the blockchain nodes;

hashing the at least one dataset to generate at least one hash value corresponding to each of the remainder of the blockchain nodes; and

determining whether the at least one hash value is stored in the blockchain node.

7. The method of claim 6, further comprising:

in response to determining that the at least one hash value is not stored in the blockchain node, determining a blockchain node that the at least one dataset corresponding to the at least one hash value is received from; and

reporting the blockchain node as a faulty node.

8. The method of claim 6, further comprising:

in response to determining that the at least one hash value corresponding to each of the remainder of the blockchain nodes is stored in the blockchain node, decoding the one or more blocks based on the at least one of the plurality of datasets stored in the blockchain node and the at least one dataset received from each of the remainder of the blockchain nodes.

9. The method of any previous claim, wherein the one or more blocks are historical blocks that have been created for a predetermined amount of time.

10. The method of any previous claim, wherein the ECC is performed when utilization rate of computational resource of the blockchain node is less than or equal to a predetermined value or usage of storage space of the blockchain node is greater than or equal to a predetermined percentage.

11. The method of any previous claim, wherein the ECC is erasure coding performed by adding redundant bits to the plurality of blocks.

12. The method of any previous claim, wherein the plurality of blocks are infrequently accessed blocks that are appended to the blockchain for a predetermined amount of time.

13. A system communicating shared blockchain data, comprising:

one or more processors; and

one or more computer-readable memories coupled to the one or more processors and having instructions stored thereon that are executable by the one or more processors to perform the method of any of claims 1 to 12.

14. An apparatus for communicating shared blockchain data, the apparatus comprising a plurality of modules for performing the method of any of claims 1 to 12.

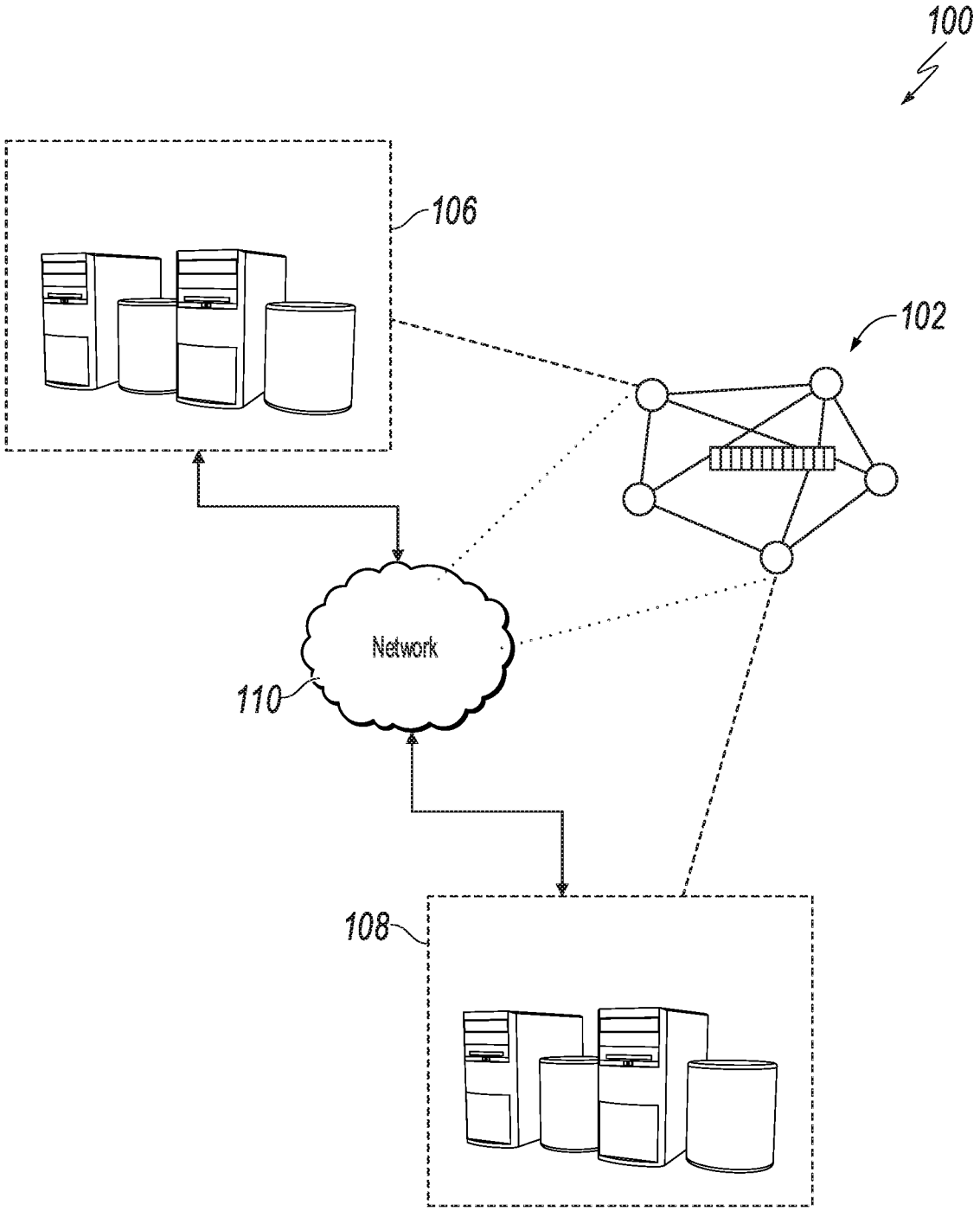


FIG. 1

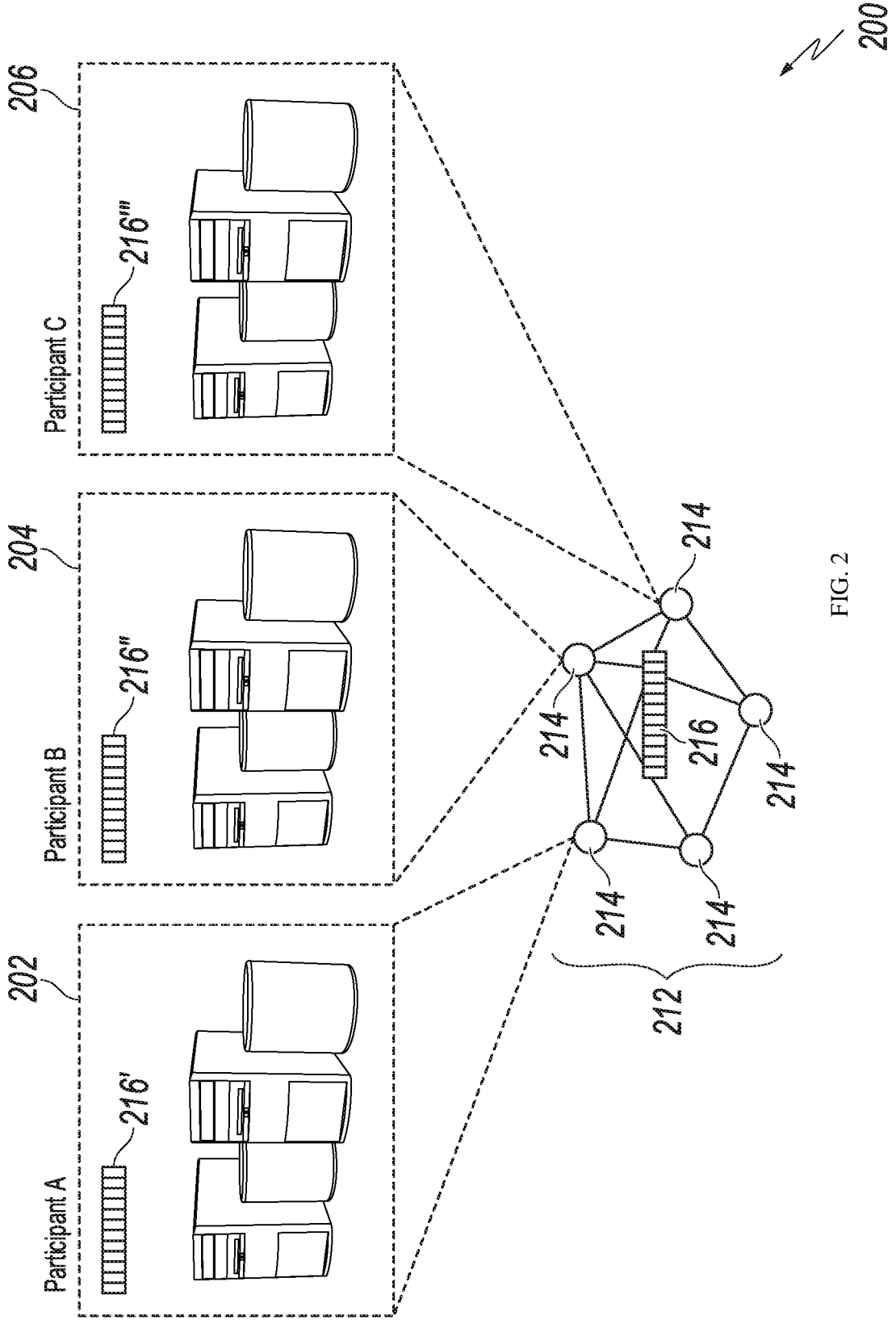
200
206
Participant C
216'''
204
Participant B
216''
202
Participant A
216'
214
214
214
214
216
212

FIG. 2

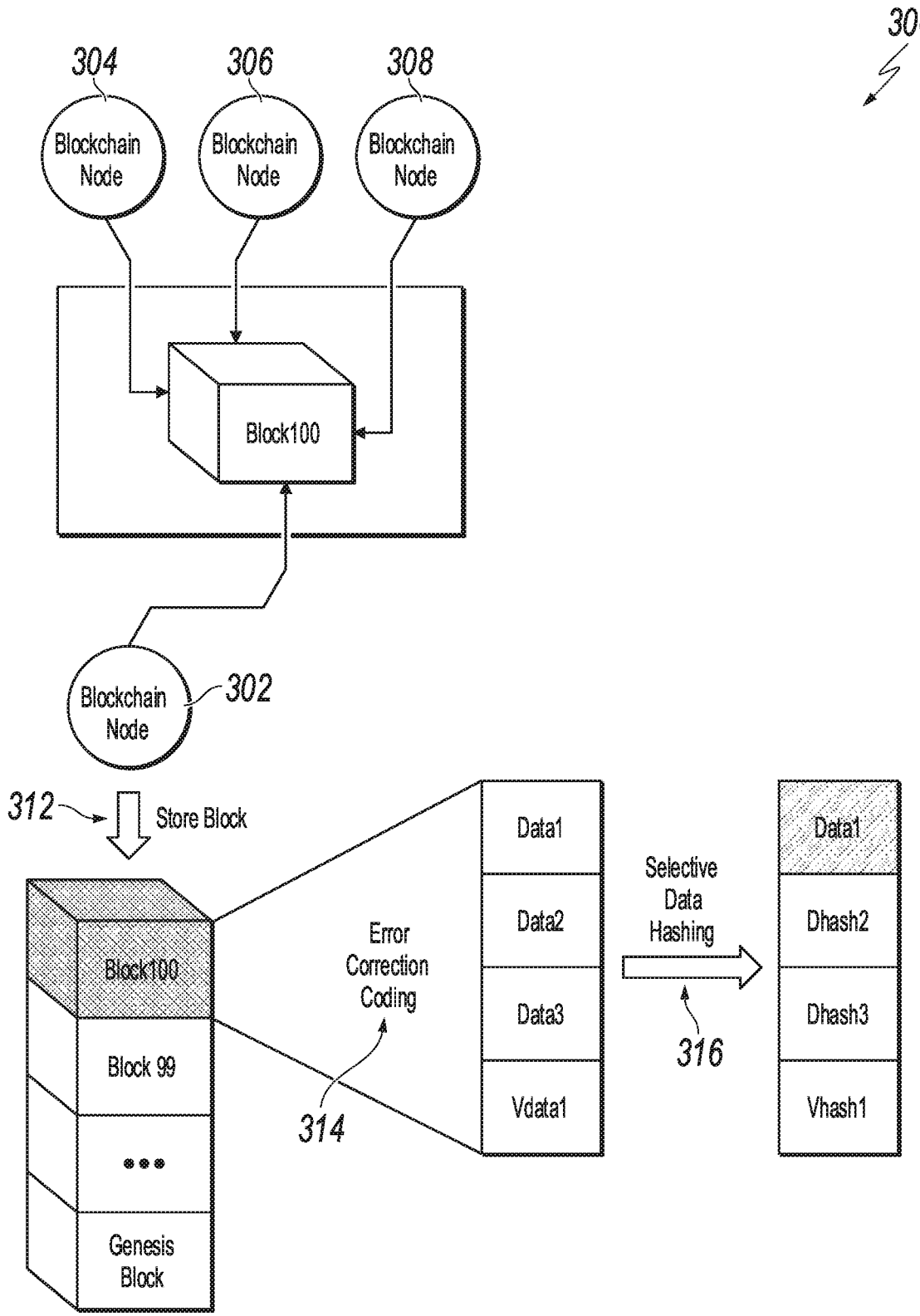
300
304
306
308
Blockchain Node
Blockchain Node
Blockchain Node
Block100
Blockchain Node
302
312
Store Block
Block100
Block 99
Genesis Block
Error Correction Coding
314
Data1
Data2
Data3
Vdata1
Selective Data Hashing
316
Data1
Dhash2
Dhash3
Vhash1

FIG. 3

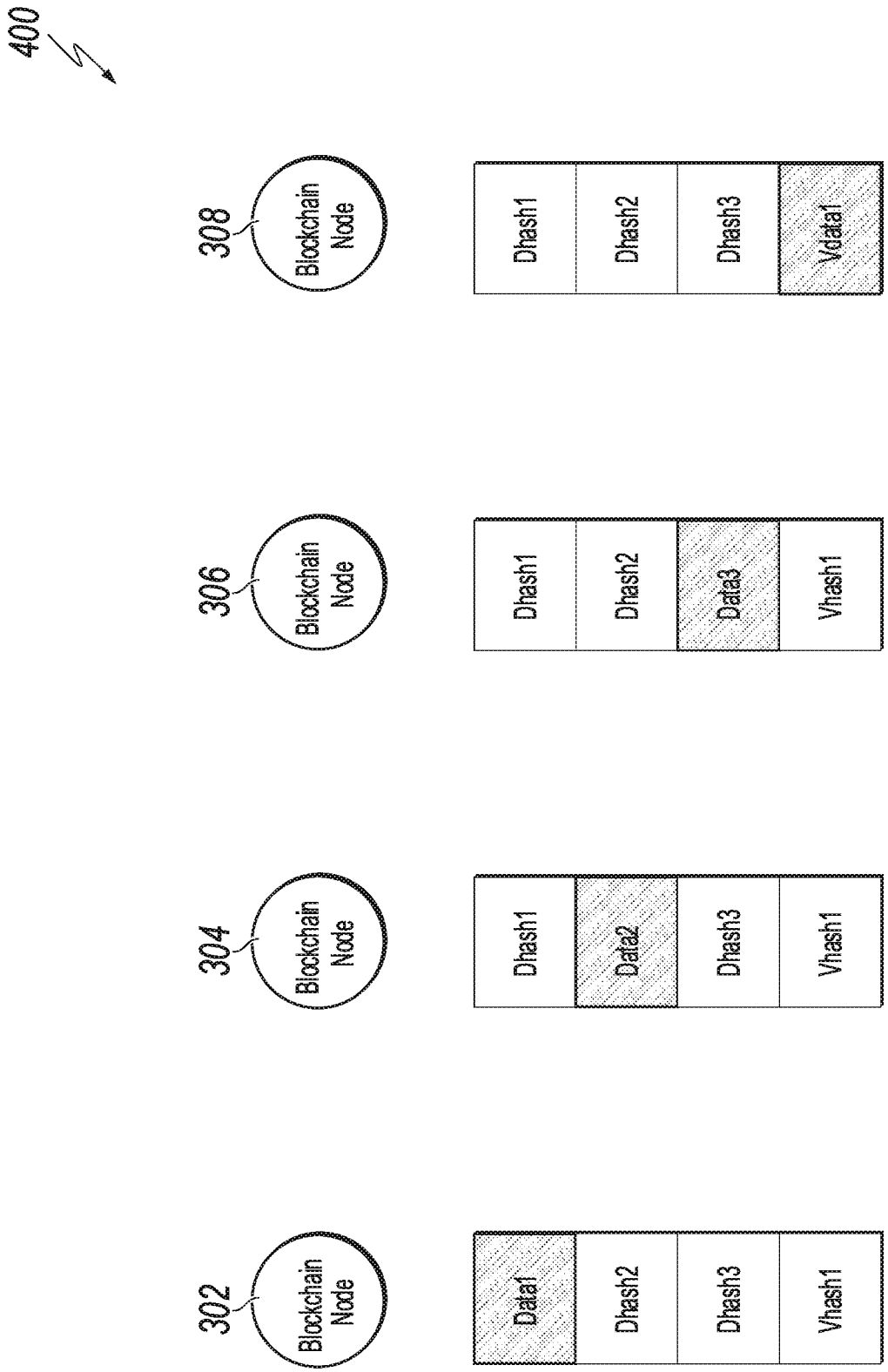
400
302
Blockchain Node
Data1
Dhash2
Dhash3
Vhash1
304
Blockchain Node
Dhash1
Data2
Dhash3
Vhash1
306
Blockchain Node
Dhash1
Dhash2
Data3
Vhash1
308
Blockchain Node
Dhash1
Dhash2
Dhash3
Vdata1

FIG. 4

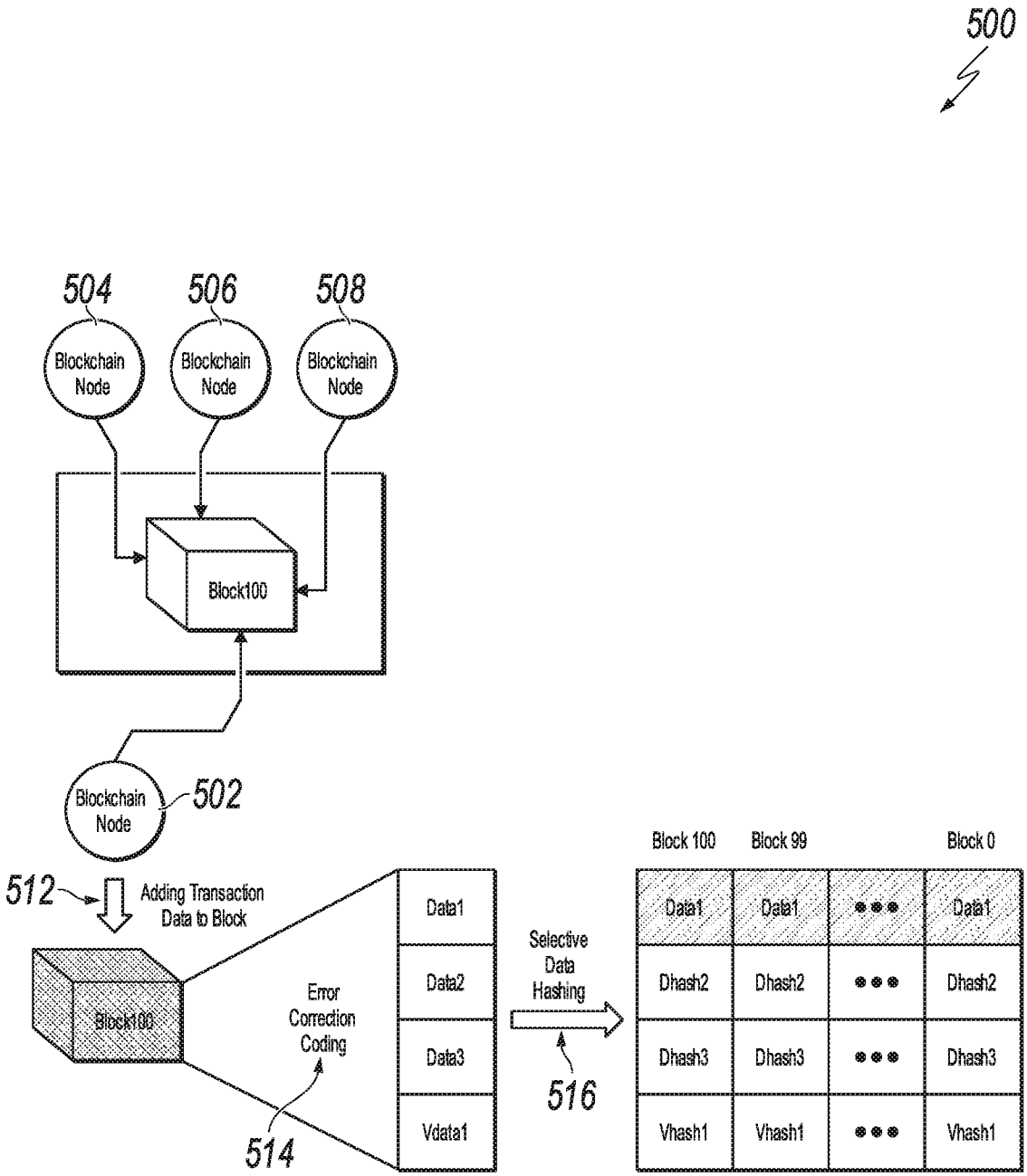
500
504
506
508
Blockchain Node
Blockchain Node
Blockchain Node
Block100
Blockchain Node
502
512
Adding Transaction Data to Block
Block100
Error Correction Coding
514
Data1
Data2
Data3
Vdata1
Selective Data Hashing
516
Block 100
Block 99
Block 0
Data1
Data1
Data1
Dhash2
Dhash2
Dhash2
Dhash3
Dhash3
Dhash3
Vhash1
Vhash1
Vhash1

FIG. 5

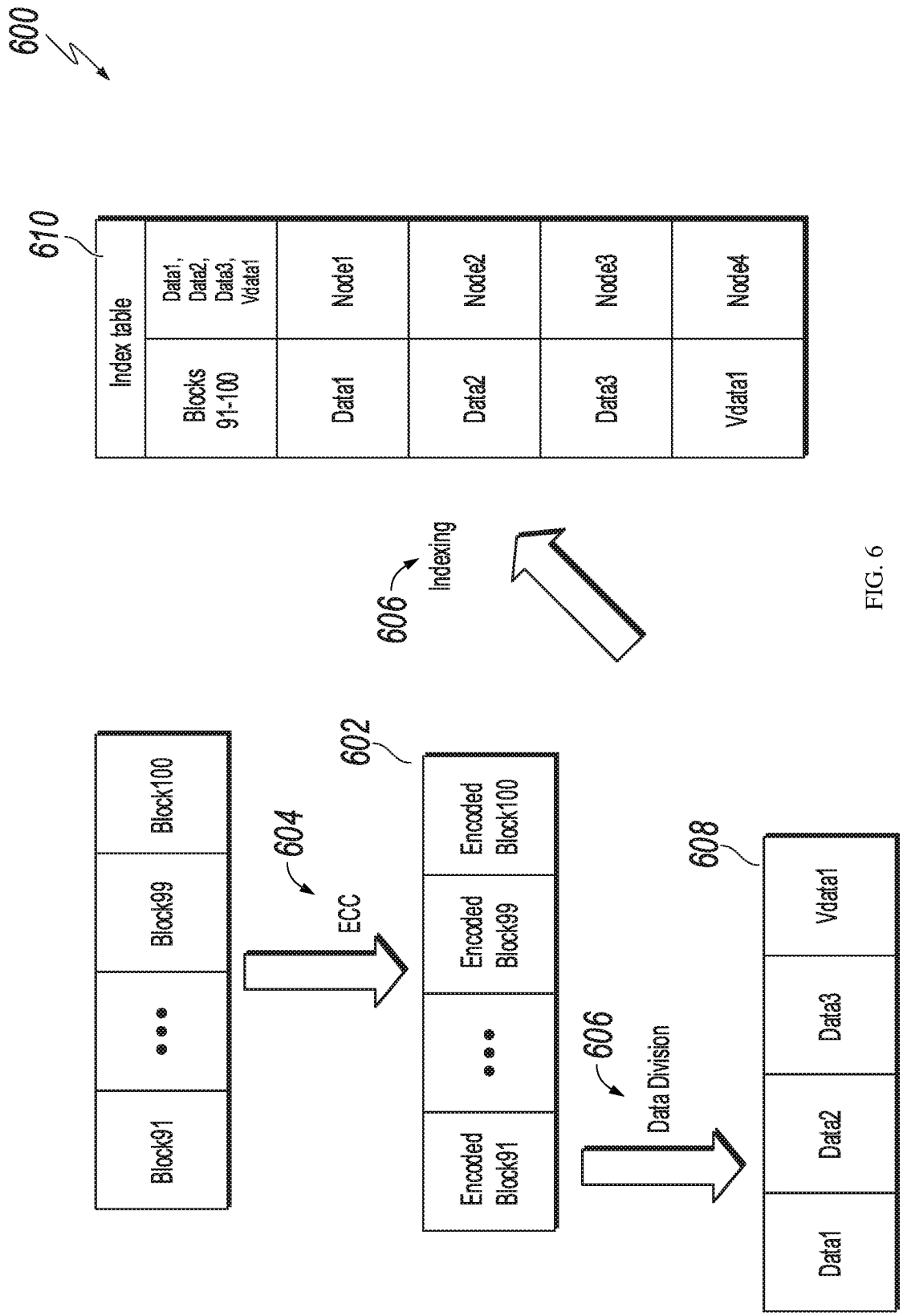
600
610
Index table
Blocks 91-100
Data1, Data2, Data3, Vdata1
Node1
Node2
Node3
Node4
Data1
Data2
Data3
Vdata1
606
Indexing
602
604
ECC
Block91
Block99
Block100
Encoded Block91
Encoded Block99
Encoded Block100
606
Data Division
608
Data1
Data2
Data3
Vdata1

FIG. 6

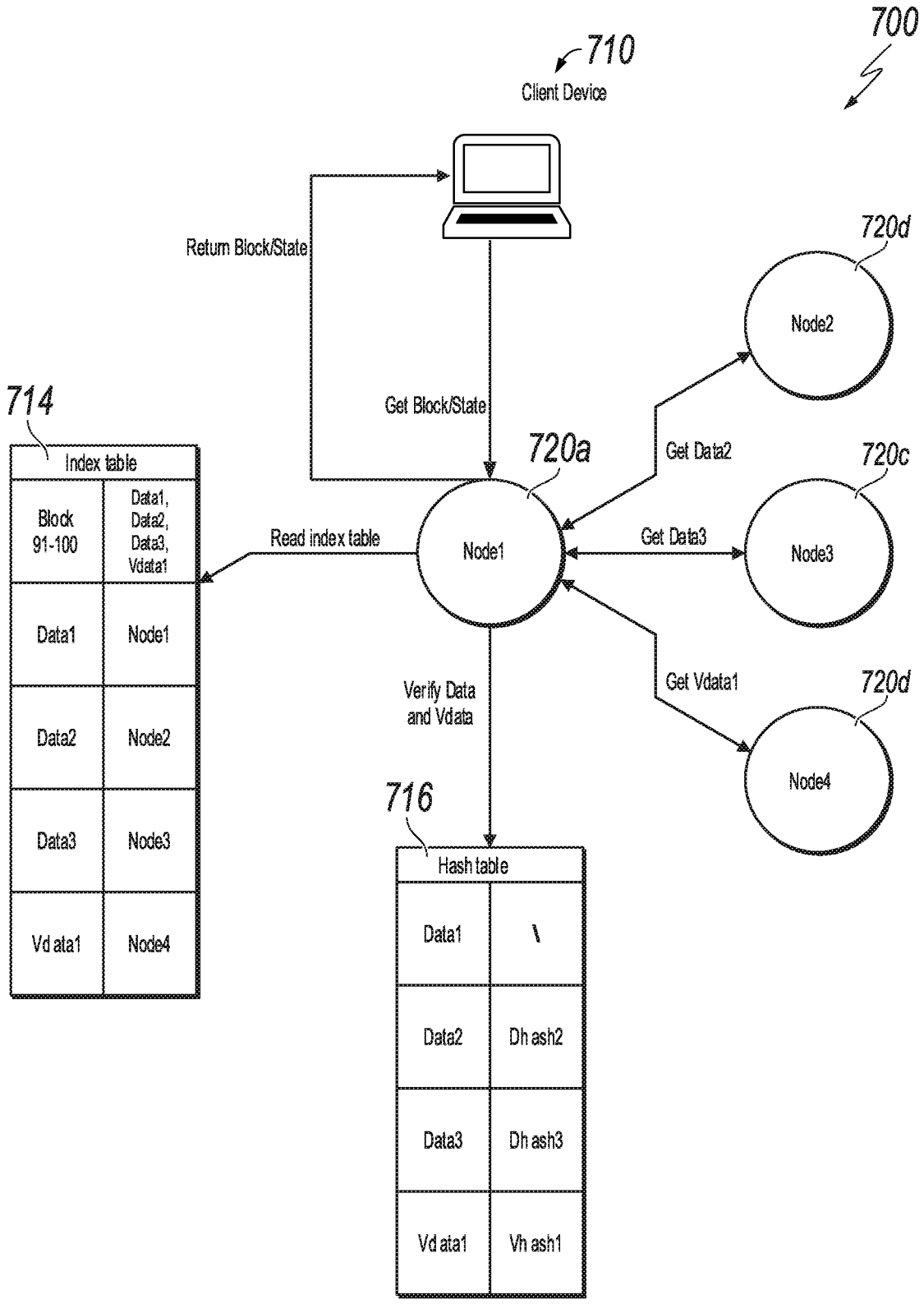
700
710
Client Device
Return Block/State
Get Block/State
720d
Node2
720a
Node1
Get Data2
720c
Node3
Get Data3
Read index table
Get Vdata1
720d
Node4
Verify Data
and Vdata
714
Index table
Block
91-100
Data1,
Data2,
Data3,
Vdata1
Data1
Node1
Data2
Node2
Data3
Node3
Vdata1
Node4
716
Hash table
Data1
\
Data2
Dhash2
Data3
Dhash3
Vdata1
Vhash1

FIG. 7

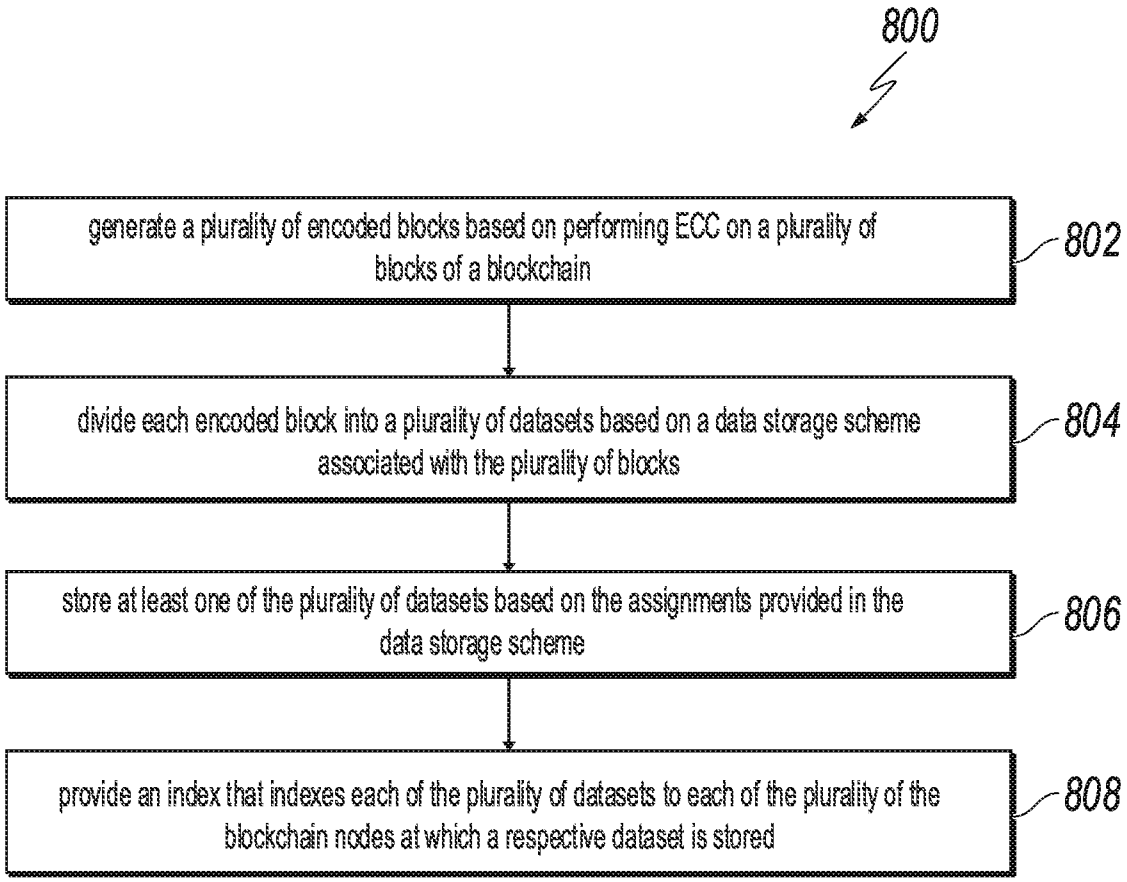
800
generate a plurality of encoded blocks based on performing ECC on a plurality of blocks of a blockchain
802
divide each encoded block into a plurality of datasets based on a data storage scheme associated with the plurality of blocks
804
store at least one of the plurality of datasets based on the assignments provided in the data storage scheme
806
provide an index that indexes each of the plurality of datasets to each of the plurality of the blockchain nodes at which a respective dataset is stored
808

FIG. 8

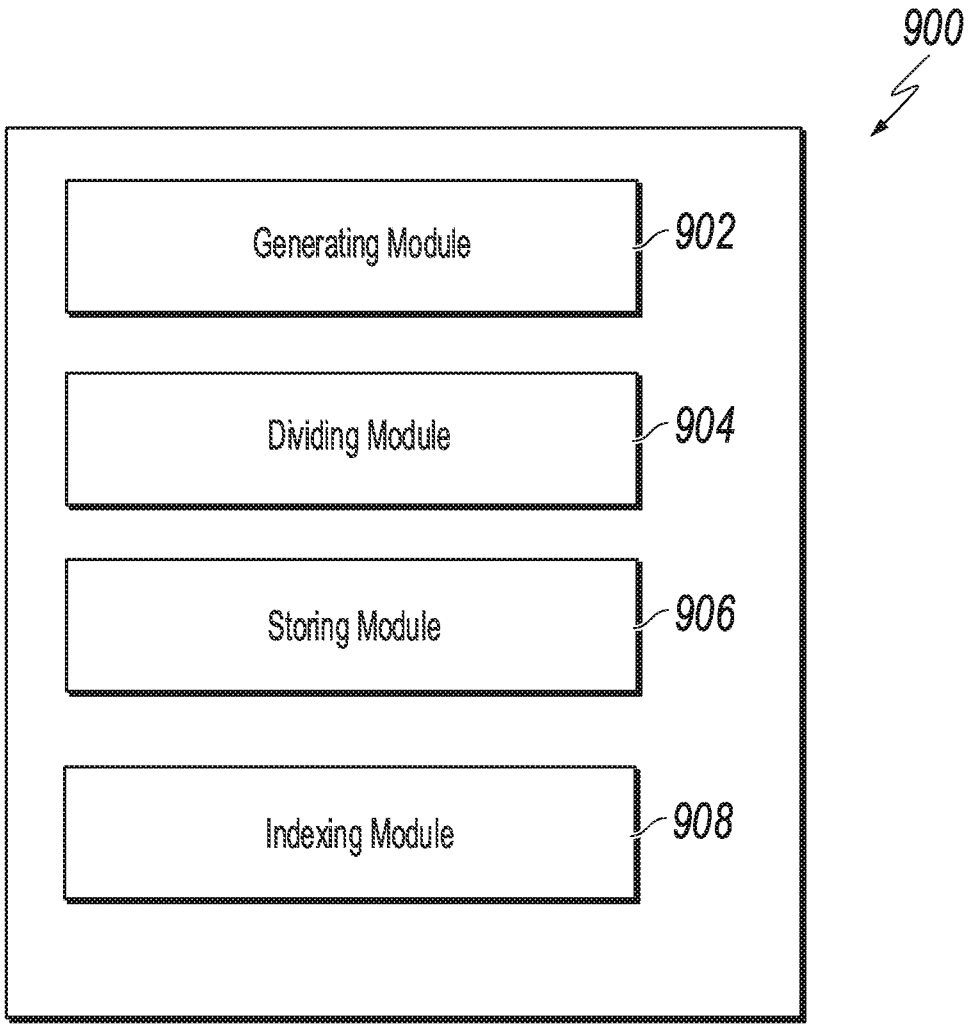
900
Generating Module
902
Dividing Module
904
Storing Module
906
Indexing Module
908

FIG. 9

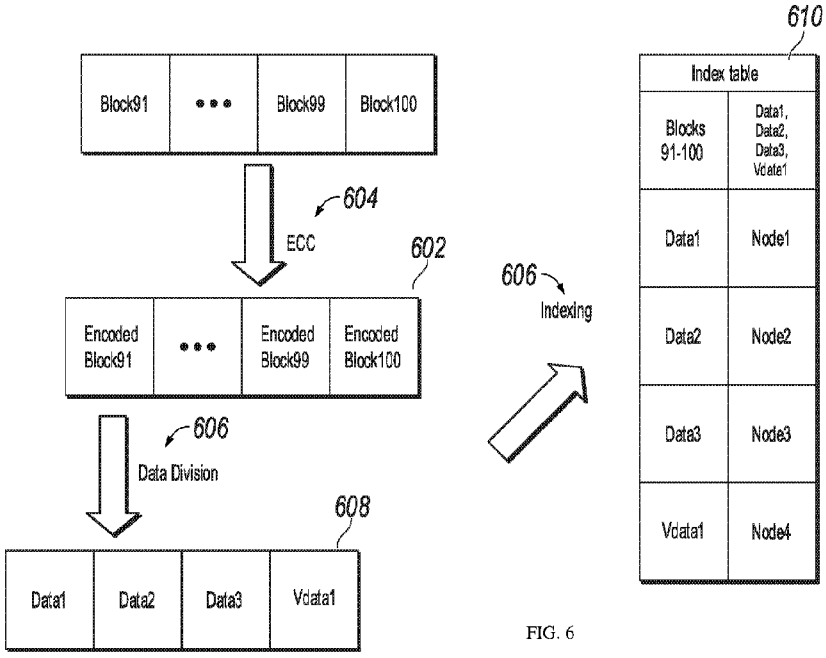
600
610
Index table
Blocks 91-100
Data1, Data2, Data3, Vdata1
Data1
Node1
Data2
Node2
Data3
Node3
Vdata1
Node4
606
Indexing
Block91
Block99
Block100
604
ECC
602
Encoded Block91
Encoded Block99
Encoded Block100
606
Data Division
608
Data1
Data2
Data3
Vdata1

FIG. 6