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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 January 2022 (13.01.2022)

WIPO | PCT

(10) International Publication Number
WO 2022/007548 A1

(51) International Patent Classification:
G06F 21/62 (2013.01)

(21) International Application Number:
PCT/CN2021/097738

(22) International Filing Date:
01 June 2021 (01.06.2021)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
16/921,325 06 July 2020 (06.07.2020) US

(71) Applicant: **INTERNATIONAL BUSINESS MACHINES CORPORATION** [US/US]; New Orchard Road, Armonk, New York 10504 (US).

(71) Applicant *(for MG only)*: **IBM (CHINA) CO., LIMITED** [CN/CN]; 7F, Bldg 10, Zhangjiang Innovation Park, 399 Keyuan Road, Zhangjiang High-Tech Campus, Pudong New Area, Shanghai 201203 (CN).

(72) Inventors: **CASELLA, Alex, Xingqi**; c/o IBM Corporation, Intellectual Property Law Department, 2455 South Road, MS P386, Poughkeepsie, New York 12601 (US). **ISHIGURO, Bonnie**; P.O. Box 218, 1101 Kitchawan Road, Yorktown Heights, New York 10598-0218 (US). **LU, Zhou**; c/o IBM Corporation, Intellectual Property Law Department, 2455 South Road, MS P386, Poughkeepsie, New York 12601 (US). **YOON, Woong Ah**; c/o IBM Corporation, Intellectual Property Law Department, 2455 South Road, MS P386, Poughkeepsie, New York 12601 (US).

(74) Agent: **KING & WOOD MALLESONS**; 20th Floor, East Tower, World Financial Centre, No. 1 Dongsanhuan Zhonglu,, Chaoyang District, Beijing 100020 (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, IT, 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, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, 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,

(54) Title: BLOCKCHAIN IMPLEMENTATION TO SECURELY STORE INFORMATION OFF-CHAIN

(57) Abstract: A processor may define a datastore connection object. The datastore connection object may include information regarding an off-chain datastore. The processor may associate the identifier to the datastore connection object. The processor may store the datastore connection object with the identifier in the blockchain network. The processor may identify that a request is being sent within the blockchain network. The request may include private information. The processor may determine whether to allow the request access to the off-chain datastore.

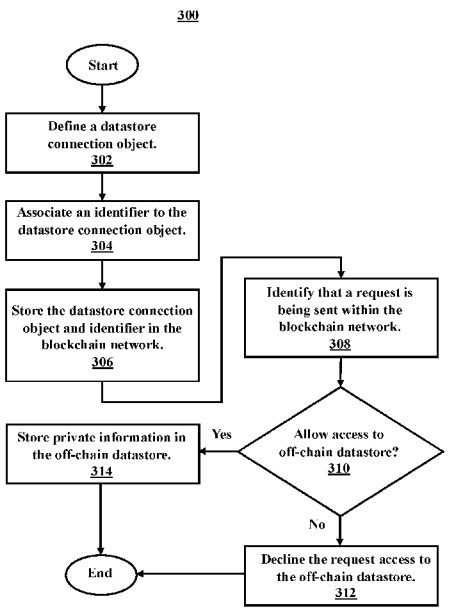

FIG. 3

WO 2022/007548 A1

[Continued on next page]

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))*

BLOCKCHAIN IMPLEMENTATION TO SECURELY STORE INFORMATION OFF-CHAIN

BACKGROUND

[0001] The present disclosure relates generally to the field of blockchain storage, and more specifically to implementing blockchain to securely store private information off-chain.

[0002] By design, every organization in a blockchain network hosts the same ledger database on its machines. Accordingly, as of now, Hyperledger Fabric Blockchain does not provide a native way to securely store and encrypt data off of the ledger (e.g., off-chain).

SUMMARY

[0003] Embodiments of the present disclosure include a method, system, and computer program product for protecting private information in a blockchain network. A processor may define a datastore connection object. The datastore connection object may include information regarding an off-chain datastore. The processor may associate the identifier to the datastore connection object. The processor may store the datastore connection object with the identifier in the blockchain network. The processor may identify that a request is being sent within the blockchain network. The request may include private information. The processor may determine whether to allow the request access to the off-chain datastore.

[0004] The above summary is not intended to describe each illustrated embodiment or every implementation of the present disclosure.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] The drawings included in the present disclosure are incorporated into, and form part of, the specification. They illustrate embodiments of the present disclosure and, along with the description, serve to explain the principles of the disclosure. The drawings are only illustrative of certain embodiments and do not limit the disclosure.

[0006] FIG. 1A illustrates an example blockchain architecture, in accordance with embodiments of the present disclosure.

[0007] FIG. 1B illustrates a blockchain transactional flow, in accordance with embodiments of the present disclosure.

[0008] FIG. 2A illustrates a block diagram of an example system where a datastore connection object is allowed access to a specific database, in accordance with embodiments of the present disclosure.

[0009] FIG. 2B illustrates a block diagram of an example network for storing private information in a single off-chain datastore, in accordance with embodiments of the present disclosure.

[0010] FIG. 3 illustrates a flowchart of an example method for protecting private information in a blockchain network, in accordance with embodiments of the present disclosure.

[0011] FIG. 4A illustrates a cloud computing environment, in accordance with embodiments of the present disclosure.

[0012] FIG. 4B illustrates abstraction model layers, in accordance with embodiments of the present disclosure.

[0013] FIG. 5 illustrates a high-level block diagram of an example computer system that may be used in implementing one or more of the methods, tools, and modules, and any related functions, described herein, in accordance with embodiments of the present disclosure.

[0014] While the embodiments described herein are amenable to various modifications and alternative forms, specifics thereof have been shown by way of example in the drawings and will be described in detail. It should be understood, however, that the particular embodiments described are not to be taken in a limiting sense. On the contrary, the intention is to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the disclosure.

DETAILED DESCRIPTION

[0015] Aspects of the present disclosure relate generally to the field of blockchain storage, and more specifically to implementing blockchain to securely store private information off-chain. As of now, Hyperledger Fabric Blockchain does not provide a native way to securely store and encrypt data off of the ledger (e.g., off-chain), and by design, every organization in a blockchain network hosts the same ledger database on its machines. When it comes to healthcare, there is not only a need to keep protected health information (e.g., private information, etc.) secure, but also to keep it in off-chain databases. Still though, healthcare providers do not always want to have their data stored on other providers' machines, even if the data is encrypted. Disclosed herein is a way to use a datastore connection object to connect to off-chain databases, store protected health data off-chain, and maintain the access control only through chaincode. In such an implementation, the storage and access control of private information is Health Insurance Portability and Accountability Act (HIPPA)-compliant.

[0016] It will be readily understood that the instant components, as generally described and illustrated in the figures herein, may be arranged and designed in a wide variety of different configurations. Accordingly, the following detailed description of the embodiments of at least one of a method, apparatus, non-transitory computer readable medium and system, as

represented in the attached figures, is not intended to limit the scope of the application as claimed but is merely representative of selected embodiments.

[0017] The instant features, structures, or characteristics as described throughout this specification may be combined or removed in any suitable manner in one or more embodiments. For example, the usage of the phrases "example embodiments," "some embodiments," or other similar language, throughout this specification refers to the fact that a particular feature, structure, or characteristic described in connection with the embodiment may be included in at least one embodiment. Accordingly, appearances of the phrases "example embodiments," "in some embodiments," "in other embodiments," or other similar language, throughout this specification do not necessarily all refer to the same group of embodiments, and the described features, structures, or characteristics may be combined or removed in any suitable manner in one or more embodiments. Further, in the FIGS., any connection between elements can permit one-way and/or two-way communication even if the depicted connection is a one-way or two-way arrow. Also, any device depicted in the drawings can be a different device. For example, if a mobile device is shown sending information, a wired device could also be used to send the information.

[0018] In addition, while the term "message" may have been used in the description of embodiments, the application may be applied to many types of networks and data. Furthermore, while certain types of connections, messages, and signaling may be depicted in exemplary embodiments, the application is not limited to a certain type of connection, message, and signaling.

[0019] Detailed herein are a method, system, and computer program product that utilize blockchain (specifically, Hyperledger Fabric) channels and smart contracts to determine whether to allow a request access to an off-chain datastore and to document the access (or denial) to the off-chain datastore.

[0020] In some embodiment, the method, system, and/or computer program product utilize a decentralized database (such as a blockchain) that is a distributed storage system, which includes multiple nodes that communicate with each other. The decentralized database may include an append-only immutable data structure resembling a distributed ledger capable of maintaining records between mutually untrusted parties. The untrusted parties are referred to herein as peers or peer nodes. Each peer maintains a copy of the database records and no single peer can modify the database records without a consensus being reached among the distributed peers. For example, the peers may execute a consensus protocol to validate blockchain storage transactions, group the storage transactions into blocks, and build a hash chain over the blocks. This process forms the ledger by ordering the storage transactions, as is necessary, for consistency.

[0021] In various embodiments, a permissioned and/or a permission-less blockchain can be used. In a public, or permission-less, blockchain, anyone can participate without a specific identity (e.g., retaining anonymity). Public blockchains can involve native cryptocurrency and use consensus based on various protocols such as Proof of Work. On the other hand, a permissioned blockchain database provides secure interactions among a group of entities which share a common goal but which do not fully trust one another, such as businesses that exchange funds, goods, (private) information, and the like.

[0022] Further, in some embodiment, the method, system, and/or computer program product can utilize a blockchain that operates arbitrary, programmable logic, tailored to a decentralized storage scheme and referred to as "smart contracts" or "chaincodes." In some cases, specialized chaincodes may exist for management functions and parameters which are referred to as system chaincode (such as managing access to an off-chain datastore/database). In some embodiments, the method, system, and/or computer program product can further utilize smart contracts that are trusted distributed applications which leverage tamper-proof

properties of the blockchain database and an underlying agreement between nodes, which is referred to as an endorsement or endorsement policy. Blockchain transactions associated with this application can be endorsed before being committed to the blockchain, while transactions, which are not endorsed, are disregarded.

[0023] An endorsement policy allows chaincode to specify endorsers for a transaction in the form of a set of peer nodes that are necessary for endorsement. When a client sends the transaction to the peers specified in the endorsement policy, the transaction is executed to validate the transaction. After validation, the transactions enter an ordering phase in which a consensus protocol is used to produce an ordered sequence of endorsed transactions grouped into blocks.

[0024] In some embodiment, the method, system, and/or computer program product can utilize nodes that are the communication entities of the blockchain system. A "node" may perform a logical function in the sense that multiple nodes of different types can run on the same physical server. Nodes are grouped in trust domains and are associated with logical entities that control them in various ways. Nodes may include different types, such as a client or submitting-client node which submits a transaction-invocation to an endorser (e.g., peer), and broadcasts transaction-proposals to an ordering service (e.g., ordering node).

[0025] Another type of node is a peer node which can receive client submitted transactions, commit the transactions, and maintain a state and a copy of the ledger of blockchain transactions. Peers can also have the role of an endorser, although it is not a requirement. An ordering-service-node or orderer is a node running the communication service for all nodes, and which implements a delivery guarantee, such as a broadcast to each of the peer nodes in the system when committing/confirming transactions (such as documenting the access of an off-chain datastore/database) and modifying a world state of the blockchain, which is another

name for the initial blockchain transaction which normally includes control and setup information.

[0026] In some embodiment, the method, system, and/or computer program product can utilize a ledger that is a sequenced, tamper-resistant record of all state transitions of a blockchain. State transitions may result from chaincode invocations (e.g., transactions) submitted by participating parties (e.g., client nodes, ordering nodes, endorser nodes, peer nodes, etc.). Each participating party (such as a peer node) can maintain a copy of the ledger. A transaction may result in a set of asset key-value pairs being committed to the ledger as one or more operands, such as creates, updates, deletes, and the like. The ledger includes a blockchain (also referred to as a chain) which is used to store an immutable, sequenced record in blocks. The ledger also includes a state database which maintains a current state of the blockchain.

[0027] In some embodiment, the method, system, and/or computer program product described herein can utilize a chain that is a transaction log that is structured as hash-linked blocks, and each block contains a sequence of N transactions where N is equal to or greater than one. The block header includes a hash of the block's transactions, as well as a hash of the prior block's header. In this way, all transactions on the ledger may be sequenced and cryptographically linked together. Accordingly, it is not possible to tamper with the ledger data without breaking the hash links. A hash of a most recently added blockchain block represents every transaction on the chain that has come before it, making it possible to ensure that all peer nodes are in a consistent and trusted state. The chain may be stored on a peer node file system (e.g., local, attached storage, cloud, etc.), efficiently supporting the append-only nature of the blockchain workload.

[0028] The current state of the immutable ledger represents the latest values for all keys that are included in the chain transaction log. Since the current state represents the latest key

values known to a channel, it is sometimes referred to as a world state. Chaincode invocations execute transactions against the current state data of the ledger. To make these chaincode interactions efficient, the latest values of the keys may be stored in a state database. The state database may be simply an indexed view into the chain's transaction log, it can therefore be regenerated from the chain at any time. The state database may automatically be recovered (or generated if needed) upon peer node startup, and before transactions are accepted.

[0029] Some benefits of the instant solutions described and depicted herein include a method, system, and computer program product for identifying discrepancies of contributions in a blockchain network. The exemplary embodiments solve the issues of time and trust by extending features of a database such as immutability, digital signatures and being a single source of truth. The exemplary embodiments provide a solution for private shared resource confirmations on blockchain. The blockchain networks may be homogenous based on the asset type (e.g., private information, cryptocurrency, etc.) and rules that govern the assets based on the smart contracts.

[0030] Blockchain is different from a traditional database in that blockchain is not a central storage, but rather a decentralized, immutable, and secure storage, where nodes may share in changes to records in the storage. Some properties that are inherent in blockchain and which help implement the blockchain include, but are not limited to, an immutable ledger, smart contracts, security, privacy, decentralization, consensus, endorsement, accessibility, and the like, which are further described herein. According to various aspects, the system described herein is implemented due to immutable accountability, security, privacy, permitted decentralization, availability of smart contracts, endorsements and accessibility that are inherent and unique to blockchain.

[0031] In particular, the blockchain ledger data is immutable, which provides for an efficient method for identifying discrepancies in a blockchain network. Also, use of the encryption in the blockchain provides security and builds trust. The smart contract manages the state of the asset to complete the life-cycle. The example blockchains are permission decentralized. Thus, each end user may have its own ledger copy to access. Multiple organizations (and peers) may be on-boarded on the blockchain network. The key organizations may serve as endorsing peers to validate the smart contract execution results, read-set and write-set. In other words, the blockchain inherent features provide for efficient implementation of processing a private transaction in a blockchain network.

[0032] One of the benefits of the example embodiments is that it improves the functionality of a computing system by implementing a method for processing a private transaction in a blockchain network (e.g., by way of protecting private information from being exposed to other peers/nodes in the blockchain network). Through the blockchain system described herein, a computing system (or a processor in the computing system) can perform functionality for private transaction processing utilizing blockchain networks by providing access to capabilities such as distributed ledger, peers, encryption technologies, MSP, event handling, etc. Also, the blockchain enables to create a business network and make any users or organizations to on-board for participation. As such, the blockchain is not just a database. The blockchain comes with capabilities to create a network of users and on-board/off-board organizations to collaborate and execute service processes in the form of smart contracts.

[0033] The example embodiments provide numerous benefits over a traditional database. For example, through the blockchain the embodiments provide for immutable accountability, security, privacy, permitted decentralization, availability of smart contracts, endorsements and accessibility that are inherent and unique to the blockchain.

[0034] Meanwhile, a traditional database could not be used to implement the example embodiments because it does not bring all parties on the network, it does not create trusted collaboration and does not provide for an efficient storage of digital assets. The traditional database does not provide for a tamper proof storage and does not provide for preservation of the digital assets being stored. Thus, the proposed embodiments described herein utilizing blockchain networks cannot be implemented in the traditional database.

[0035] Meanwhile, if a traditional database were to be used to implement the example embodiments, the example embodiments would have suffered from unnecessary drawbacks such as search capability, lack of security and slow speed of transactions. Accordingly, the example embodiments provide for a specific solution to a problem in the arts/field of private information storage and/or processing.

[0036] The example embodiments also change how data may be stored within a block structure of the blockchain. For example, a digital asset data may be securely stored within a certain portion of the data block (e.g., an identifier associated with an off-chain connection object stored within a header, data segment, metadata, etc.). By storing the digital asset data within data blocks of a blockchain, the digital asset data may be appended to an immutable blockchain ledger through a hash-linked chain of blocks. In some embodiments, the data block may be different than a traditional data block by having a personal data/private information associated with the digital asset not stored together with the assets within a traditional block structure of a blockchain. By removing the personal data/private information associated with the digital asset, the blockchain can provide the benefit of anonymity based on immutable accountability and security.

[0037] Turning now to FIG. 1A, illustrated is a blockchain architecture 100, in accordance with embodiments of the present disclosure. In some embodiments, the blockchain architecture 100 may include certain blockchain elements, for example, a group of blockchain

nodes 102. The blockchain nodes 102 may include one or more blockchain nodes, e.g., peers 104-110 (these four nodes are depicted by example only). These nodes participate in a number of activities, such as a blockchain transaction addition and validation process (consensus). One or more of the peers 104-110 may endorse and/or recommend transactions based on an endorsement policy and may provide an ordering service for all blockchain nodes 102 in the blockchain architecture 100. A blockchain node may initiate a blockchain authentication and seek to write to a blockchain immutable ledger stored in blockchain layer 116, a copy of which may also be stored on the underpinning physical infrastructure 114. The blockchain configuration may include one or more applications 124 which are linked to application programming interfaces (APIs) 122 to access and execute stored program/application code 120 (e.g., chaincode, smart contracts, etc.) which can be created according to a customized configuration sought by participants and can maintain their own state, control their own assets, and receive external information. This can be deployed as a transaction and installed, via appending to the distributed ledger, on all blockchain nodes 104-110.

[0038] The blockchain base or platform 112 may include various layers of blockchain data, services (e.g., cryptographic trust services, virtual execution environment, etc.), and underpinning physical computer infrastructure that may be used to receive and store new transactions and provide access to auditors which are seeking to access data entries. The blockchain layer 116 may expose an interface that provides access to the virtual execution environment necessary to process the program code and engage the physical infrastructure 114. Cryptographic trust services 118 may be used to verify transactions such as asset exchange transactions and keep information private.

[0039] The blockchain architecture 100 of FIG. 1A may process and execute program/application code 120 via one or more interfaces exposed, and services provided, by

blockchain platform 112. The application code 120 may control blockchain assets. For example, the application code 120 can store and transfer data, and may be executed by peers 104-110 in the form of a smart contract and associated chaincode with conditions or other code elements subject to its execution. As a non-limiting example, smart contracts may be created to execute the transfer of resources, the generation of resources, etc. The smart contracts can themselves be used to identify rules associated with authorization, access requirements (e.g., of a datastore, an off-chain datastore, etc.), and/or usage of the ledger. For example, the off-chain datastore information 126 may be processed by one or more processing entities (e.g., virtual machines) included in the blockchain layer 116. The result 128 may include a plurality of linked shared documents (e.g., with each linked shared document recording the issuance of a smart contract in regard to the off-chain datastore information 126 being identified as either being allowed or denied access to an off-chain datastore, etc.). In some embodiments, the physical infrastructure 114 may be utilized to retrieve any of the data or information described herein.

[0040] A smart contract may be created via a high-level application and programming language, and then written to a block in the blockchain. The smart contract may include executable code which is registered, stored, and/or replicated with a blockchain (e.g., a distributed network of blockchain peers). A transaction is an execution of the smart contract code that can be performed in response to conditions associated with the smart contract being satisfied. The executing of the smart contract may trigger a trusted modification(s) to a state of a digital blockchain ledger. The modification(s) to the blockchain ledger caused by the smart contract execution may be automatically replicated throughout the distributed network of blockchain peers through one or more consensus protocols.

[0041] The smart contract may write data to the blockchain in the format of key-value pairs. Furthermore, the smart contract code can read the values stored in a blockchain and use them

in application operations. The smart contract code can write the output of various logic operations into the blockchain. The code may be used to create a temporary data structure in a virtual machine or other computing platform. Data written to the blockchain can be public and/or can be encrypted and maintained as private. The temporary data that is used/generated by the smart contract is held in memory by the supplied execution environment, then deleted once the data needed for the blockchain is identified.

[0042] A chaincode may include the code interpretation of a smart contract, with additional features. As described herein, the chaincode may be program code deployed on a computing network, where it is executed and validated by chain validators together during a consensus process. The chaincode receives a hash and retrieves from the blockchain a hash associated with the data template created by use of a previously stored feature extractor. If the hashes of the hash identifier and the hash created from the stored identifier template data match, then the chaincode sends an authorization key to the requested service. The chaincode may write to the blockchain data associated with the cryptographic details (e.g., thus confirming a contribution, identifying a discrepancy with a contribution, etc.).

[0043] FIG. 1B illustrates an example of a blockchain transactional flow 150 between nodes of the blockchain in accordance with an example embodiment. Referring to FIG. 1B, the transaction flow may include a transaction proposal 191 sent by an application client node 160 to an endorsing peer node 181 (e.g., in some embodiments, the transaction proposal 191 may be a request that includes an identifier associated with an off-chain datastore/database). The endorsing peer 181 may verify the client signature and execute a chaincode function to initiate the transaction. The output may include the chaincode results, a set of key/value versions that were read in the chaincode (read set), and the set of keys/values that were written in chaincode (write set). The proposal response 192 is sent back to the client 160 along with an endorsement signature, if approved. The client 160 assembles the endorsements

into a transaction payload 193 and broadcasts it to an ordering service node 184. The ordering service node 184 then delivers ordered transactions as blocks to all peers 181-183 on a channel. Before committal to the blockchain, each peer 181-183 may validate the transaction. For example, the peers may check the endorsement policy to ensure that the correct allotment of the specified peers have signed the results and authenticated the signatures against the transaction payload 193 (e.g., all, or a threshold number of peers, validate that the request includes the identifier and/or symmetric key that allows the finding of an datastore connection object and/or access to an off-chain database).

[0044] Referring again to FIG. 1B, the client node 160 initiates the transaction 191 by constructing and sending a request to the peer node 181, which is an endorser. The client 160 may include an application leveraging a supported software development kit (SDK), which utilizes an available API to generate a transaction proposal 191. The proposal is a request to invoke a chaincode function so that data can be read and/or written to the ledger (e.g., write new key value pairs for the assets). The SDK may reduce the package of the transaction proposal 191 into a properly architected format (e.g., protocol buffer over a remote procedure call (RPC)) and take the client's cryptographic credentials to produce a unique signature for the transaction proposal 191.

[0045] In response, the endorsing peer node 181 may verify (a) that the transaction proposal 191 is well formed, (b) the transaction has not been submitted already in the past (replay-attack protection), (c) the signature is valid, and (d) that the submitter (client 160, in the example) is properly authorized to perform the proposed operation on that channel. The endorsing peer node 181 may take the transaction proposal 191 inputs as arguments to the invoked chaincode function. The chaincode is then executed against a current state database to produce transaction results including a response value, read set, and write set. However, no updates are made to the ledger at this point. In some embodiments, the set of values, along

with the endorsing peer node's 181 signature is passed back as a proposal response 192 to the SDK of the client 160 which parses the payload for the application to consume.

[0046] In response, the application of the client 160 inspects/verifies the endorsing peers signatures and compares the proposal responses to determine if the proposal response is the same. If the chaincode only queried the ledger, the application would inspect the query response and would typically not submit the transaction to the ordering node service 184. If the client application intends to submit the transaction to the ordering node service 184 to update the ledger, the application determines if the specified endorsement policy has been fulfilled before submitting (e.g., has a request been accepted). Here, the client may include only one of multiple parties to the transaction. In this case, each client may have their own endorsing node, and each endorsing node will need to endorse the transaction. The architecture is such that even if an application selects not to inspect responses or otherwise forwards an unendorsed transaction, the endorsement policy will still be enforced by peers and upheld at the commit validation phase.

[0047] After successful inspection, in the transaction payload step 193, the client 160 assembles endorsements into a transaction and broadcasts the transaction proposal 191 and response within a transaction message to the ordering node 184. The transaction may contain the read/write sets, the endorsing peers signatures and a channel ID. The ordering node 184 does not need to inspect the entire content of a transaction in order to perform its operation, instead the ordering node 184 may simply receive transactions from all channels in the network, order them chronologically by channel, and create blocks of transactions per channel.

[0048] The blocks of the transaction are delivered from the ordering node 184 to all peer nodes 181-183 on the channel. The transactions 194 within the block are validated to ensure any endorsement policy is fulfilled and to ensure that there have been no changes to ledger

state for read set variables since the read set was generated by the transaction execution. Transactions in the block are tagged as being valid or invalid. Furthermore, in steps 195 each peer node 181-183 appends the block to the channel's chain, and for each valid transaction the write sets are committed to current state database. An event is emitted, to notify the client application that the transaction (invocation) has been immutably appended to the chain, as well as to notify whether the transaction was validated or invalidated (e.g., whether the request is allowed, or denied, access to an off-chain datastore).

[0049] Referring now to FIG. 2A, illustrated is a block diagram of an example system 200 where a datastore connection object 203 is allowed access to a specific database 208B, in accordance with embodiments of the present disclosure. As depicted, the system 200 includes a blockchain network 202, and off-chain datastores 206A-N. In some embodiments, the blockchain network 202 is a Hyperledger fabric blockchain network maintained by blockchain peers as discussed above in regard to FIGS. 1A-B. Further, it is noted that only three off-chain datastores 206A-N are depicted, but any number of datastores are contemplated for purposes of this disclosure.

[0050] In some embodiments, the blockchain network 202 includes the datastore connection object 203 which includes an off-chain datastore information 204. In some embodiments, each off-chain datastore 206A-N includes respective databases 208A-N. Again, as with the off-chain datastores 206A-N, although each off-chain datastore 206A-N is depicted as respectively including two databases 208A-N the off-chain datastores 206A-N may include any number of databases (e.g., off-chain datastore 206A includes five database, off-chain datastore 206B includes one database, etc.).

[0051] As depicted, and proposed herein, is the system 200 (and a method and a computer program product) that stores encrypted, private (health) information to one or more of the off-chain datastores 206A-N through the datastore connection object 203 which is stored in the

blockchain network 202. In such a way, the storage and handling of the private information is compliant with HIPPA. In some embodiments, the datastore connection object 203 contains off-chain data information 204, which is needed to connect to one of the off-chain datastores 206A-N and points to a database 208A-N within the datastores 206A-N (which as depicted is connecting to database 208B within off-chain datastore 206A). In some embodiments, the off-chain datastore connection information 204 is encrypted on a blockchain ledger, and access to any of the off-chain datastores 206A-N is managed through chaincode. In such a proposed design, as depicted in regard to system 200 of FIG. 2A, it is ensured that only users/entities/peers/etc. with proper access can read and/or write from a datastore 206A-N, thus providing data security, integrity, and privacy.

[0052] In some embodiments, the solution proposed in regard to FIG. 2A manages create, read, update, and delete (CRUD) operations for connecting and storing private information (e.g., protected health data) to a database 208A-N of an off-chain datastore 206A-N. As disclosed, private data of an asset included in a request, etc. (such as any protected health information) is stored off-chain with the help of the datastore connection object 203. In some embodiments, a chaincode may developer define the datastore connection object 203, which contains off-chain datastore information 204 such the datastore's host, an account username and/or password, and a specific database (e.g., 208B) within a datastore (e.g., 206A). In some embodiments, off-chain datastore information 204 is specified in a "connection string" field of the datastore connection object 203, for example, in URL-encoded form. In some embodiments, the datastore connection object 203 is then encrypted with a symmetric key and saved to a ledger of the blockchain network 202.

[0053] In some embodiments, for each desired off-chain datastore 206A-N type (e.g., Cloudant by IBM, etc.), a chaincode developer implements the logic for how the datastore connection object 203 or datastore connection objects respectively associated with each off-

chain datastore 206A-N will be parsed and how the parsed information (e.g., including the off-chain datastore information 204) of the datastore connection object 203 will be used to execute operations such as put, get, and delete against databases of that type (e.g., databases 208A-B of off-chain datastore 206A, etc.). This could involve, for example, constructing an HTTP request given the information in the datastore connection object 203 and the asset data supplied by the transaction. Such a mechanism allows an application to connect to any type of off-chain datastore 206A-N that is outside of the Hyperledger Fabric network.

[0054] In some embodiments, once the aforementioned steps are complete, the datastore connection object 203 can be referenced by an identifier in an asset's metadata, which is stored on the ledger (e.g., a request with the identifier is directed to the datastore connection object 203 via the identifier and is subsequently allowed or denied access to an off-chain datastore 206A-N that may house private information in a database 208A-N). In some embodiments, an asset's private information is encrypted with a symmetric key, and if a datastore connection identifier is specified (e.g., within the asset and/or request associated with the asset), the private data is stored on a corresponding off-chain datastore (e.g., 206A-N), as opposed to the ledger.

[0055] It is noted that since the datastore connection object 203 is encrypted on the ledger, only users/entities who are given access to the symmetric key (associated with the datastore connection object 203) will be able to access the connection information associated with a corresponding off-chain datastore 206A-N and attempt to retrieve the asset's private information (e.g., access and/or store the private information from an asset associated with a request into a designated database 208A-N or provide private information from a designated database 208A-N in response to the asset). In some embodiments, this access is managed by the chaincode. In some embodiments, in order to give a user access to the datastore

connection object 203, an RSA public key associated with a user/entity is used to encrypt the symmetric key of the datastore connection object 203.

[0056] Referring now to FIG. 2B, illustrated is a block diagram of an example network 220 for storing private information in a single off-chain datastore 238, in accordance with embodiments of the present disclosure. In some embodiments, the system 203 includes application code 222A-N, chaincode 224A-N, and the off-chain datastore 238. In some embodiments, the chaincode 224A-N respectively includes ledgers 226A-N and the off-chain datastore 238 includes databases 240A-N. It is noted that although there are set number of elements 222-240 in the FIG. 2B, it is envisioned that there may be any amount of elements used.

[0057] In some embodiments, as depicted, there can be multiple databases 240A-B within a single off-chain datastore 238. This means that multiple chaincode 224A-N solutions can use the same off-chain datastore 238 while keeping their (the chaincode 224A-N) data isolated from one another. Such an embodiment allows chaincode (e.g., 224A-N) developers the flexibility to decide how to allocate databases (e.g., 240A-B) for blockchain solutions running on the same network. As depicted, such a setup involves allocating a either database 240A, or database 240B, to each solution so that the entire network 200 uses only one datastore 238.

[0058] It is further noted that the disclosed system 200 is flexible in that it allows chaincode (223A-N) developers to provide their own databases (e.g., perhaps 240A-B, or others). In such a flexible embodiment, the setup steps are: defining a datastore connection object (such as 203 of FIG. 2A) for each off-chain database (such as 240 and/or 208A-N of FIG. 2A); instantiating each off-chain databases; and, for every off-chain datastore (e.g., 238 and/or 206A-N of FIG. 2A) type, implementing logic for executing operations against the databases using the datastore connection information. In some embodiments, a datastore

connection object can be created at any time during an application's lifetime (using application code 222A-N), which means that additional databases can be added dynamically with ease.

[0059] In some embodiments, for access management purposes, a user's RSA keys and asset symmetric keys are to be stored on a blockchain ledger, herein depicted as the ledgers 226A-N (however, it is noted that the ledgers 226A-N are the same blockchain ledger and include the same or substantially the same transaction information). In some embodiments, all CRUD operations on the datastore connection object are first validated by checking whether a request/caller has direct, or indirect, access to the object's key. In such an embodiment, each operation can be logged on the ledgers 226-N for auditing purposes.

[0060] In some embodiments, since invoke transactions are executed by each endorsing peer, further disclosed herein is a mechanism that avoids writing the same asset (e.g., private [health] information]) to an off-chain datastore 224A-N multiple times. The mechanism involves each off-chain datastore (e.g., 230 and/or 206A-N) being key-value datastores, and a hash of the asset data is saved as the key. Thus, before saving asset data, the hash is computed and checked to see if the key already exists in a datastore (e.g., indicating that the asset already exists in the datastore). In some embodiments, the hash is further used to maintain data integrity, that is, when retrieving the asset data, hash is computed and compared to the above-mentioned key to check whether they match. If the hash and the key match, there is confidence that the asset data has not been tampered with.

[0061] In some embodiments, since what is disclosed herein allows for multiple off-chain datastore connection objects for the same datastore 238, unique off-chain datastore objects for each set of user/entity credentials can be generated. It is noted that this allows different users (with their own associated identifiers and passwords for a datastore) to be given off-chain access without compromising each other's security. For example, a first hospital may

have access to database 240A on the off-chain datastore 238 by having a unique identifier embedded in requests asking for patient information. The unique identifier is used as a key to validated that the requests came from the owning entity, e.g., the first hospital and the request is allowed to gather the patient information. Whereas, a second hospital may have access to database 240B by having another unique identifier and only requests with the another unique identifier would be allowed to gather information from database 240B. Further, it is noted that both first hospital and the second hospital are barred from accessing the other's respectively associated database 240A and 240B.

[0062] Referring now to FIG. 3, illustrated is a flowchart of an example method 300 for protecting private information in a blockchain network, in accordance with embodiments of the present disclosure. In some embodiments, the method 300 may be performed by a processor. In some embodiments, the processor may be a part of a blockchain (e.g., Hyperledger Fabric) network.

[0063] In some embodiments, the method 300 begins at operation 302. At operation 302, the processor defines a[n off-chain] datastore connection object. The datastore connection object includes information regarding an off-chain datastore. In some embodiments, the method 300 proceeds to operation 304, where the processor associates an identifier to the datastore connection object.

[0064] In some embodiments, the method 300 proceeds to operation 306, where the processor stores the datastore connection object and the identifier in the blockchain network (e.g., in the Hyperledger Fabric). In some embodiments, the method 300 proceeds to operation 308. At operation 308, the processor identifies that a request (e.g., for a transaction, as a transaction, with an asset, etc.) is being sent within the blockchain network. In some embodiments, the request includes private information.

[0065] In some embodiments, the method 300 proceeds to decision block 310, where it is determined whether to allow the request access to the off-chain datastore. In some embodiments, if at decision block 310, it is identified that the request does not include the identifier, the method 300 proceeds to operation 312. At operation 312, the processor declines the request access to the off-chain datastore. In some embodiments, if it is identified that the request does not include the identifier, the processor declines the request the location of the datastore connection object, which effectively declines the request access to the off-chain datastore.

[0066] In some embodiments that are not depicted in the method 300, the processor further encrypts the datastore connection object with a symmetric key. In such an embodiment, the method 300 may proceed to operation 314, if it is identified that the request includes the identifier and the symmetric key. At operation 314, the processor may store the private information in the off-chain datastore. In some embodiments, to store the private information in the off-chain datastore, the processor may direct the request to the datastore connection object in the blockchain network and allow the request access to the off-chain datastore. In some embodiments, after either operation 312 and/or operation 314, the method 300 ends.

[0067] In some embodiments, discussed below, there are one or more operations of the method 300 not depicted for the sake of brevity. Accordingly, in some embodiments, storing the private information in the off-chain datastore in operation 314, may further include the processor computing a hash of the private information and identifying that the has does not exist in the off-chain datastore.

[0068] In some embodiments, the processor may (further) identify that a second request is being sent within the blockchain network. The second request may include the private information. In some embodiments, the processor computes a hash of the private information and identifies that the hash already exists in the off-chain datastore. Upon identifying that the

hash already exists in the off-chain datastore, the processor may decline the storing of the private information in the off-chain datastore. In some embodiments, as discussed above, if it is identified that the has does not already exist in the off-chain datastore, the private information is stored in the off-chain datastore.

[0069] It is noted that, in some embodiments, the off-chain datastore may house the private information and the private information may include health data associated with one or more users. Further noted is, in some embodiments, the information regarding the off-chain datastore includes an identity of a host of the datastore, an account username and/or password (associated with the identity of the host), and/or a location of a specific database in the datastore (which houses/locates the private information).

[0070] It is to be understood that although this disclosure includes a detailed description on cloud computing, implementation of the teachings recited herein are not limited to a cloud computing environment. Rather, embodiments of the present disclosure are capable of being implemented in conjunction with any other type of computing environment now known or later developed.

[0071] Cloud computing is a model of service delivery for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, network bandwidth, servers, processing, memory, storage, applications, virtual machines, and services) that can be rapidly provisioned and released with minimal management effort or interaction with a provider of the service. This cloud model may include at least five characteristics, at least three service models, and at least four deployment models.

[0072] Characteristics are as follows:

[0073] On-demand self-service: a cloud consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with the service's provider.

[0074] Broad network access: capabilities are available over a network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs).

[0075] Resource pooling: the provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to demand. There is a sense of portion independence in that the consumer generally has no control or knowledge over the exact portion of the provided resources but may be able to specify portion at a higher level of abstraction (e.g., country, state, or datacenter).

[0076] Rapid elasticity: capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.

[0077] Measured service: cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

[0078] Service Models are as follows:

[0079] Software as a Service (SaaS): the capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through a thin client interface such as a web browser (e.g., web-based e-mail). The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application

capabilities, with the possible exception of limited user-specific application configuration settings.

[0080] Platform as a Service (PaaS): the capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including networks, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations.

[0081] Infrastructure as a Service (IaaS): the capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls).

[0082] Deployment Models are as follows:

[0083] Private cloud: the cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on-premises or off-premises.

[0084] Community cloud: the cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be managed by the organizations or a third party and may exist on-premises or off-premises.

[0085] Public cloud: the cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.

[0086] Hybrid cloud: the cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by

standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load-balancing between clouds).

[0087] A cloud computing environment is service oriented with a focus on statelessness, low coupling, modularity, and semantic interoperability. At the heart of cloud computing is an infrastructure that includes a network of interconnected nodes.

[0088] FIG. 4A, illustrated is a cloud computing environment 410 is depicted. As shown, cloud computing environment 410 includes one or more cloud computing nodes 400 with which local computing devices used by cloud consumers, such as, for example, personal digital assistant (PDA) or cellular telephone 400A, desktop computer 400B, laptop computer 400C, and/or automobile computer system 400N may communicate. Nodes 400 may communicate with one another. They may be grouped (not shown) physically or virtually, in one or more networks, such as Private, Community, Public, or Hybrid clouds as described hereinabove, or a combination thereof.

[0089] This allows cloud computing environment 410 to offer infrastructure, platforms and/or software as services for which a cloud consumer does not need to maintain resources on a local computing device. It is understood that the types of computing devices 400A-N shown in FIG. 4A are intended to be illustrative only and that computing nodes 400 and cloud computing environment 410 can communicate with any type of computerized device over any type of network and/or network addressable connection (e.g., using a web browser).

[0090] FIG. 4B, illustrated is a set of functional abstraction layers provided by cloud computing environment 410 (FIG. 4A) is shown. It should be understood in advance that the components, layers, and functions shown in FIG. 4B are intended to be illustrative only and embodiments of the disclosure are not limited thereto. As depicted below, the following layers and corresponding functions are provided.

[0091] Hardware and software layer 415 includes hardware and software components. Examples of hardware components include: mainframes 402; RISC (Reduced Instruction Set Computer) architecture based servers 404; servers 406; blade servers 408; storage devices 411; and networks and networking components 412. In some embodiments, software components include network application server software 414 and database software 416.

[0092] Virtualization layer 420 provides an abstraction layer from which the following examples of virtual entities may be provided: virtual servers 422; virtual storage 424; virtual networks 426, including virtual private networks; virtual applications and operating systems 428; and virtual clients 430.

[0093] In one example, management layer 440 may provide the functions described below. Resource provisioning 442 provides dynamic procurement of computing resources and other resources that are utilized to perform tasks within the cloud computing environment. Metering and Pricing 444 provide cost tracking as resources are utilized within the cloud computing environment, and billing or invoicing for consumption of these resources. In one example, these resources may include application software licenses. Security provides identity verification for cloud consumers and tasks, as well as protection for data and other resources. User portal 446 provides access to the cloud computing environment for consumers and system administrators. Service level management 448 provides cloud computing resource allocation and management such that required service levels are met. Service Level Agreement (SLA) planning and fulfillment 450 provide pre-arrangement for, and procurement of, cloud computing resources for which a future requirement is anticipated in accordance with an SLA.

[0094] Workloads layer 460 provides examples of functionality for which the cloud computing environment may be utilized. Examples of workloads and functions which may be provided from this layer include: mapping and navigation 462; software development and

lifecycle management 464; virtual classroom education delivery 466; data analytics processing 468; transaction processing 470; and private channel communication 472.

[0095] FIG. 5, illustrated is a high-level block diagram of an example computer system 501 that may be used in implementing one or more of the methods, tools, and modules, and any related functions, described herein (e.g., using one or more processor circuits or computer processors of the computer), in accordance with embodiments of the present disclosure. In some embodiments, the major components of the computer system 501 may comprise one or more CPUs 502, a memory subsystem 504, a terminal interface 512, a storage interface 516, an I/O (Input/Output) device interface 514, and a network interface 518, all of which may be communicatively coupled, directly or indirectly, for inter-component communication via a memory bus 503, an I/O bus 508, and an I/O bus interface unit 510.

[0096] The computer system 501 may contain one or more general-purpose programmable central processing units (CPUs) 502A, 502B, 502C, and 502D, herein generically referred to as the CPU 502. In some embodiments, the computer system 501 may contain multiple processors typical of a relatively large system; however, in other embodiments the computer system 501 may alternatively be a single CPU system. Each CPU 502 may execute instructions stored in the memory subsystem 504 and may include one or more levels of on-board cache.

[0097] System memory 504 may include computer system readable media in the form of volatile memory, such as random access memory (RAM) 522 or cache memory 524. Computer system 501 may further include other removable/non-removable, volatile/non-volatile computer system storage media. By way of example only, storage system 526 can be provided for reading from and writing to a non-removable, non-volatile magnetic media, such as a "hard drive." Although not shown, a magnetic disk drive for reading from and writing to a removable, non-volatile magnetic disk (e.g., a "floppy disk"), or an optical disk drive for

reading from or writing to a removable, non-volatile optical disc such as a CD-ROM, DVD-ROM or other optical media can be provided. In addition, memory 504 can include flash memory, e.g., a flash memory stick drive or a flash drive. Memory devices can be connected to memory bus 503 by one or more data media interfaces. The memory 504 may include at least one program product having a set (e.g., at least one) of program modules that are configured to carry out the functions of various embodiments.

[0098] One or more programs/utilities 528, each having at least one set of program modules 530 may be stored in memory 504. The programs/utilities 528 may include a hypervisor (also referred to as a virtual machine monitor), one or more operating systems, one or more application programs, other program modules, and program data. Each of the operating systems, one or more application programs, other program modules, and program data or some combination thereof, may include an implementation of a networking environment. Programs 528 and/or program modules 530 generally perform the functions or methodologies of various embodiments.

[0099] Although the memory bus 503 is shown in FIG. 5 as a single bus structure providing a direct communication path among the CPUs 502, the memory subsystem 504, and the I/O bus interface 510, the memory bus 503 may, in some embodiments, include multiple different buses or communication paths, which may be arranged in any of various forms, such as point-to-point links in hierarchical, star or web configurations, multiple hierarchical buses, parallel and redundant paths, or any other appropriate type of configuration. Furthermore, while the I/O bus interface 510 and the I/O bus 508 are shown as single respective units, the computer system 501 may, in some embodiments, contain multiple I/O bus interface units 510, multiple I/O buses 508, or both. Further, while multiple I/O interface units are shown, which separate the I/O bus 508 from various communications paths running to the various

I/O devices, in other embodiments some or all of the I/O devices may be connected directly to one or more system I/O buses.

[0100] In some embodiments, the computer system 501 may be a multi-user mainframe computer system, a single-user system, or a server computer or similar device that has little or no direct user interface, but receives requests from other computer systems (clients). Further, in some embodiments, the computer system 501 may be implemented as a desktop computer, portable computer, laptop or notebook computer, tablet computer, pocket computer, telephone, smartphone, network switches or routers, or any other appropriate type of electronic device.

[0101] It is noted that FIG. 5 is intended to depict the representative major components of an exemplary computer system 501. In some embodiments, however, individual components may have greater or lesser complexity than as represented in FIG. 5, components other than or in addition to those shown in FIG. 5 may be present, and the number, type, and configuration of such components may vary.

[0102] As discussed in more detail herein, it is contemplated that some or all of the operations of some of the embodiments of methods described herein may be performed in alternative orders or may not be performed at all; furthermore, multiple operations may occur at the same time or as an internal part of a larger process.

[0103] The present disclosure may be a system, a method, and/or a computer program product at any possible technical detail level of integration. The computer program product may include a computer readable storage medium (or media) having computer readable program instructions thereon for causing a processor to carry out aspects of the present disclosure.

[0104] The computer readable storage medium can be a tangible device that can retain and store instructions for use by an instruction execution device. The computer readable storage

medium may be, for example, but is not limited to, an electronic storage device, a magnetic storage device, an optical storage device, an electromagnetic storage device, a semiconductor storage device, or any suitable combination of the foregoing. A non-exhaustive list of more specific examples of the computer readable storage medium includes the following: a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), a static random access memory (SRAM), a portable compact disc read-only memory (CD-ROM), a digital versatile disk (DVD), a memory stick, a floppy disk, a mechanically encoded device such as punch-cards or raised structures in a groove having instructions recorded thereon, and any suitable combination of the foregoing. A computer readable storage medium, as used herein, is not to be construed as being transitory signals *per se*, such as radio waves or other freely propagating electromagnetic waves, electromagnetic waves propagating through a waveguide or other transmission media (e.g., light pulses passing through a fiber-optic cable), or electrical signals transmitted through a wire.

[0105] Computer readable program instructions described herein can be downloaded to respective computing/processing devices from a computer readable storage medium or to an external computer or external storage device via a network, for example, the Internet, a local area network, a wide area network and/or a wireless network. The network may comprise copper transmission cables, optical transmission fibers, wireless transmission, routers, firewalls, switches, gateway computers and/or edge servers. A network adapter card or network interface in each computing/processing device receives computer readable program instructions from the network and forwards the computer readable program instructions for storage in a computer readable storage medium within the respective computing/processing device.

[0106] Computer readable program instructions for carrying out operations of the present disclosure may be assembler instructions, instruction-set-architecture (ISA) instructions, machine instructions, machine dependent instructions, microcode, firmware instructions, state-setting data, configuration data for integrated circuitry, or either source code or object code written in any combination of one or more programming languages, including an object oriented programming language such as Smalltalk, C++, or the like, and procedural programming languages, such as the "C" programming language or similar programming languages. The computer readable program instructions may execute entirely on the user's computer, partly on the user's computer, as a stand-alone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the user's computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the Internet using an Internet Service Provider). In some embodiments, electronic circuitry including, for example, programmable logic circuitry, field-programmable gate arrays (FPGA), or programmable logic arrays (PLA) may execute the computer readable program instructions by utilizing state information of the computer readable program instructions to personalize the electronic circuitry, in order to perform aspects of the present disclosure.

[0107] Aspects of the present disclosure are described herein with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems), and computer program products according to embodiments of the disclosure. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer readable program instructions.

[0108] These computer readable program instructions may be provided to a processor of a computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks. These computer readable program instructions may also be stored in a computer readable storage medium that can direct a computer, a programmable data processing apparatus, and/or other devices to function in a particular manner, such that the computer readable storage medium having instructions stored therein comprises an article of manufacture including instructions which implement aspects of the function/act specified in the flowchart and/or block diagram block or blocks.

[0109] The computer readable program instructions may also be loaded onto a computer, other programmable data processing apparatus, or other device to cause a series of operational steps to be performed on the computer, other programmable apparatus or other device to produce a computer implemented process, such that the instructions which execute on the computer, other programmable apparatus, or other device implement the functions/acts specified in the flowchart and/or block diagram block or blocks.

[0110] The flowchart and block diagrams in the Figures illustrate the architecture, functionality, and operation of possible implementations of systems, methods, and computer program products according to various embodiments of the present disclosure. In this regard, each block in the flowchart or block diagrams may represent a module, segment, or portion of instructions, which comprises one or more executable instructions for implementing the specified logical function(s). In some alternative implementations, the functions noted in the blocks may occur out of the order noted in the Figures. For example, two blocks shown in succession may, in fact, be accomplished as one step, executed concurrently, substantially concurrently, in a partially or wholly temporally overlapping manner, or the blocks may

sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block of the block diagrams and/or flowchart illustration, and combinations of blocks in the block diagrams and/or flowchart illustration, can be implemented by special purpose hardware-based systems that perform the specified functions or acts or carry out combinations of special purpose hardware and computer instructions.

[0111] The descriptions of the various embodiments of the present disclosure have been presented for purposes of illustration, but are not intended to be exhaustive or limited to the embodiments disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art without departing from the scope and spirit of the described embodiments. The terminology used herein was chosen to best explain the principles of the embodiments, the practical application or technical improvement over technologies found in the marketplace, or to enable others of ordinary skill in the art to understand the embodiments disclosed herein.

[0112] Although the present disclosure has been described in terms of specific embodiments, it is anticipated that alterations and modification thereof will become apparent to the skilled in the art. Therefore, it is intended that the following claims be interpreted as covering all such alterations and modifications as fall within the true spirit and scope of the disclosure.

WHAT IS CLAIMED IS:

1. A method for protecting private information in a blockchain network, the method comprising:

defining a datastore connection object, wherein the datastore connection object includes information regarding an off-chain datastore;

associating an identifier to the datastore connection object;

storing the datastore connection object with the identifier in the blockchain network;

identifying that a request is being sent within the blockchain network, wherein the request includes private information;

determining whether to allow the request access to the off-chain datastore.

2. The method of claim 1, wherein the off-chain datastore houses the private information, and wherein the private information includes health data associated with one or more users.

3. The method of claim 1, wherein the information regarding the off-chain datastore includes an identity of a host of the datastore, an account username and password, and a location of a specific database in the datastore.

4. The method of claim 1, wherein determining whether to allow the request access to the off-chain datastore comprises:

identifying that the request does not include the identifier; and

declining the request access to the off-chain datastore.

5. The method of claim 1, further comprising:

encrypting the datastore connection object with a symmetric key.

6. The method of claim 5, wherein determining whether to allow the request access to the off-chain datastore comprises:

identifying that the request includes the identifier and the symmetric key;

directing the request to the datastore connection object in the blockchain network;

allowing the request to access the off-chain datastore; and

storing the private information in the off-chain datastore.

7. The method of claim 6, wherein storing the private information in the off-chain datastore comprises:

computing a hash of the private information; and

identifying that the hash does not exist in the off-chain datastore.

8. The method of claim 6, further comprising:

identifying that a second request is being sent within the blockchain network, wherein the second request includes the private information;

computing a hash of the private information;

identifying that the hash already exists in the off-chain datastore; and

declining to store the private information in the off-chain datastore.

9. A system for protecting private information in a blockchain network, the system comprising:

a memory; and

a processor in communication with the memory, the processor being configured to perform operations comprising:

defining a datastore connection object, wherein the datastore connection object includes information regarding an off-chain datastore;

associating an identifier to the datastore connection object;

storing the datastore connection object with the identifier in the blockchain network;

identifying that a request is being sent within the blockchain network, wherein the request includes private information;

determining whether to allow the request access to the off-chain datastore.

10. The system of claim 9, wherein the off-chain datastore houses the private information, and wherein the private information includes health data associated with one or more users.

11. The system of claim 9, wherein the information regarding the off-chain datastore includes an identity of a host of the datastore, an account username and password, and a location of a specific database in the datastore.

12. The system of claim 10, wherein determining whether to allow the request access to the off-chain datastore comprises:

identifying that the request does not include the identifier; and

declining the request access to the off-chain datastore.

13. The system of claim 9, wherein the operations further comprise:

encrypting the datastore connection object with a symmetric key.

14. The system of claim 13, wherein determining whether to allow the request access to the off-chain datastore comprises:

identifying that the request includes the identifier and the symmetric key;

directing the request to the datastore connection object in the blockchain network;

allowing the request to access the off-chain datastore; and

storing the private information in the off-chain datastore.

15. The system of claim 14, wherein storing the private information in the off-chain datastore comprises:

computing a hash of the private information; and

identifying that the hash does not exist in the off-chain datastore.

16. The system of claim 14, wherein the operations further comprise:

identifying that a second request is being sent within the blockchain network, wherein the second request includes the private information;

computing a hash of the private information;

identifying that the hash already exists in the off-chain datastore; and

declining to store the private information in the off-chain datastore.

17. A computer program product for protecting private information in a blockchain network, the computer program product comprising a computer readable storage medium having program instructions embodied therewith, the program instructions executable by a processor to cause the processors to perform a function, the function comprising:

defining a datastore connection object, wherein the datastore connection object includes information regarding an off-chain datastore;

associating an identifier to the datastore connection object;

storing the datastore connection object with the identifier in the blockchain network;

identifying that a request is being sent within the blockchain network, wherein the request includes private information;

determining whether to allow the request access to the off-chain datastore.

18. The computer program product of claim 17, further comprising:

encrypting the datastore connection object with a symmetric key.

19. The computer program product of claim 18, wherein determining whether to allow the request access to the off-chain datastore comprises:

identifying that the request includes the identifier and the symmetric key;

directing the request to the datastore connection object in the blockchain network;

allowing the request to access the off-chain datastore; and

storing the private information in the off-chain datastore.

20. The computer program product of claim 19, further comprising:

identifying that a second request is being sent within the blockchain network, wherein the second request includes the private information;

computing a hash of the private information;

identifying that the hash already exists in the off-chain datastore; and

declining to store the private information in the off-chain datastore.

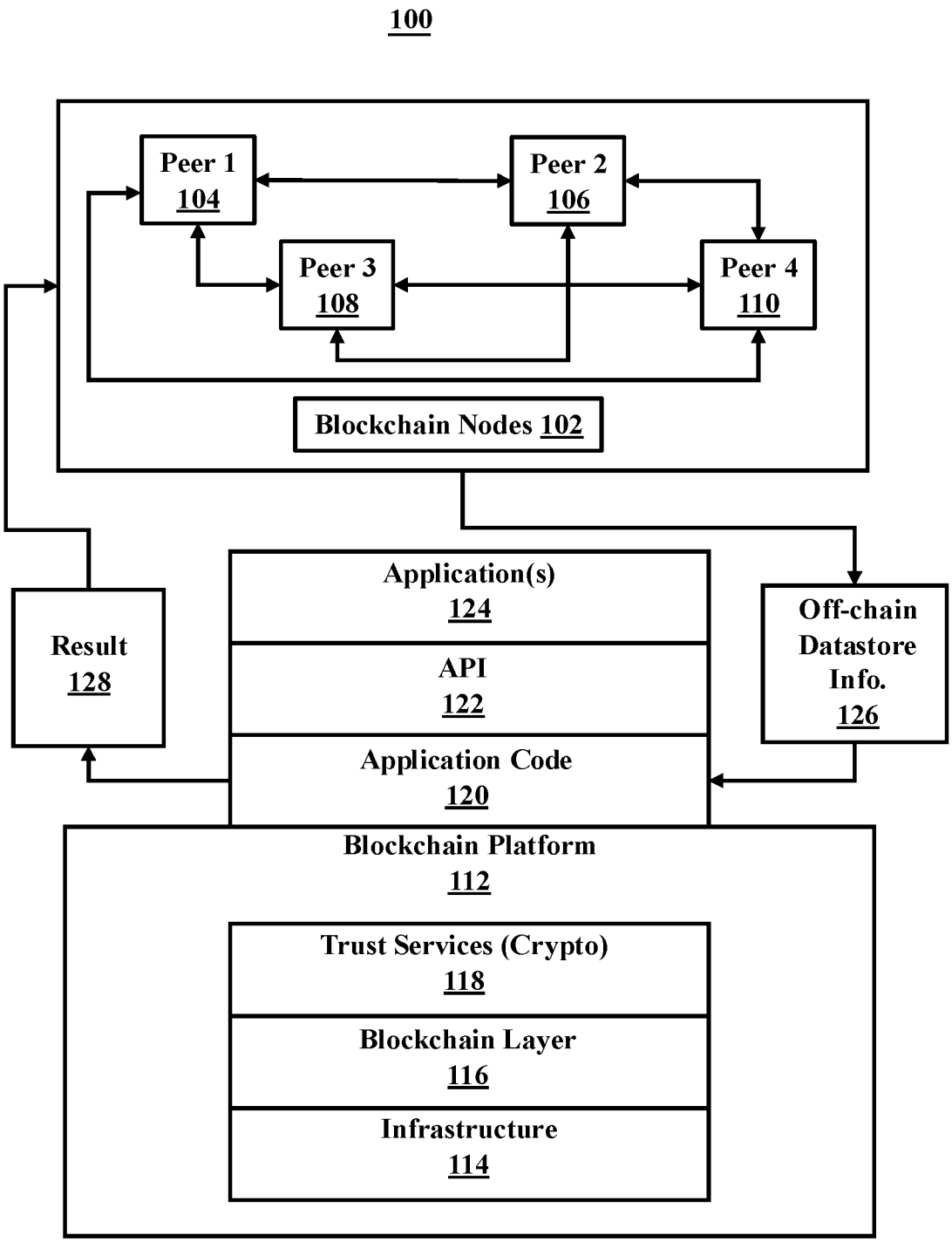
100
Peer 1
104
Peer 2
106
Peer 3
108
Peer 4
110
Blockchain Nodes 102
Result
128
Application(s)
124
API
122
Application Code
120
Off-chain
Datastore
Info.
126
Blockchain Platform
112
Trust Services (Crypto)
118
Blockchain Layer
116
Infrastructure
114

FIG. 1A

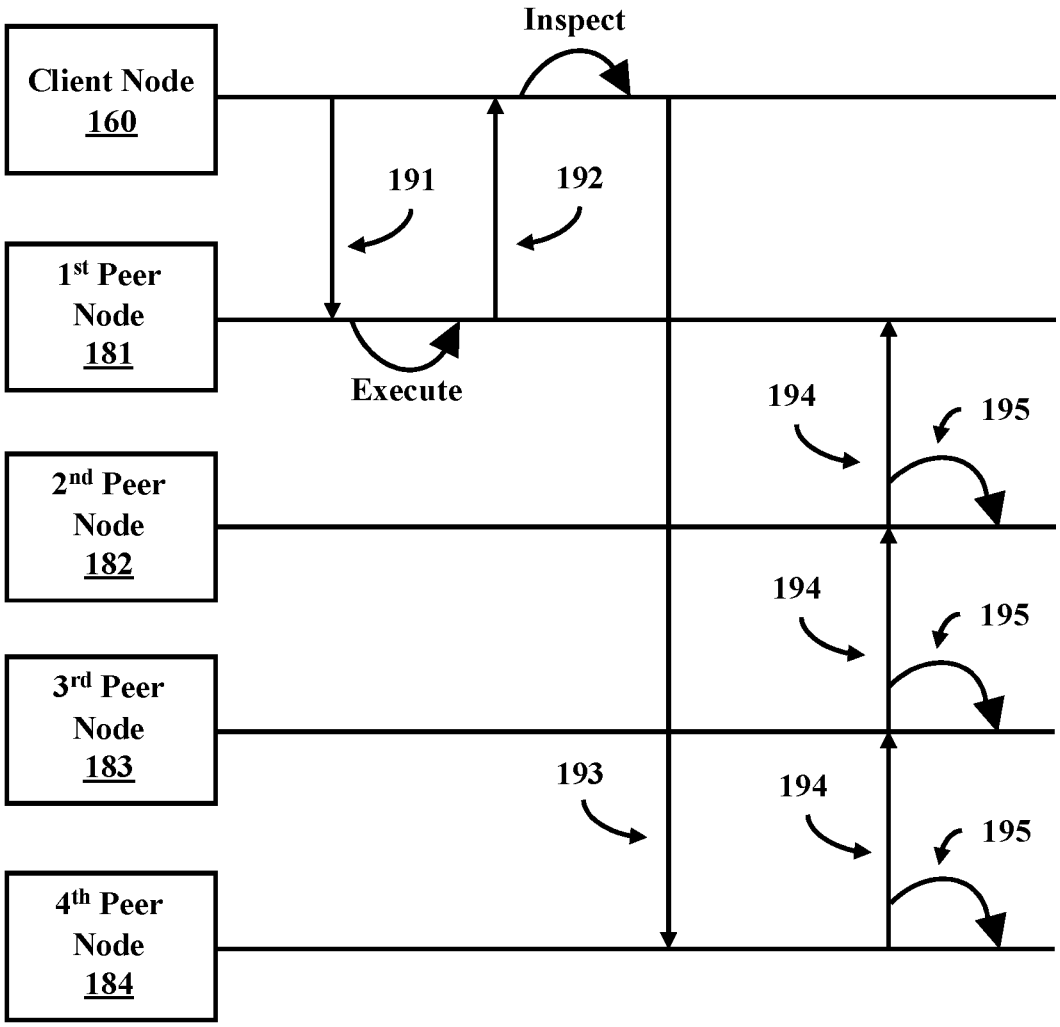
150
Client Node
160
1st Peer
Node
181
2nd Peer
Node
182
3rd Peer
Node
183
4th Peer
Node
184
Inspect
191
192
Execute
193
194
195
194
195
194
195

FIG. 1B

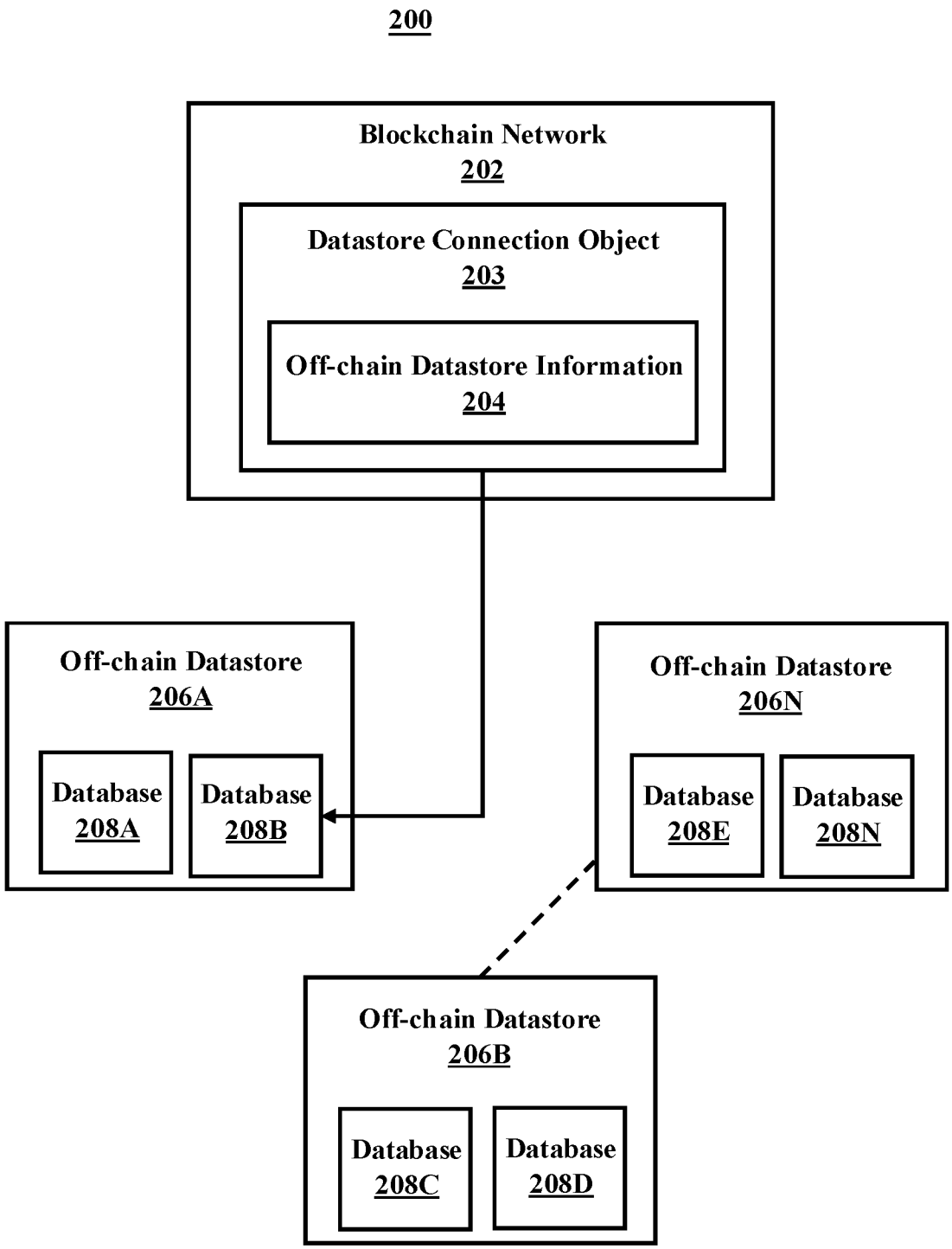
200
Blockchain Network
202
Datastore Connection Object
203
Off-chain Datastore Information
204
Off-chain Datastore
206A
Database
208A
Database
208B
Off-chain Datastore
206N
Database
208E
Database
208N
Off-chain Datastore
206B
Database
208C
Database
208D

FIG. 2A

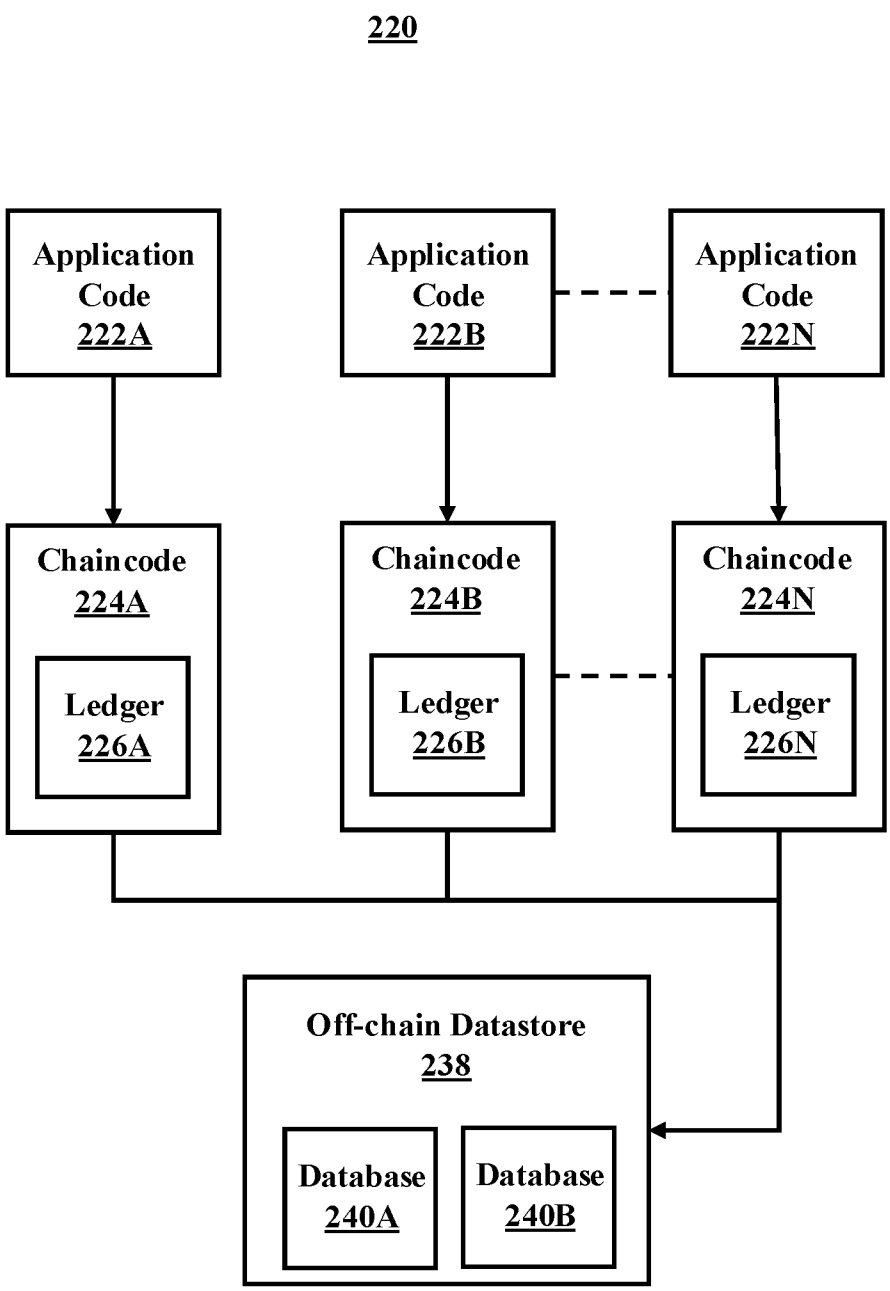
220
Application Code 222A
Application Code 222B
Application Code 222N
Chaincode 224A
Ledger 226A
Chaincode 224B
Ledger 226B
Chaincode 224N
Ledger 226N
Off-chain Datastore 238
Database 240A
Database 240B

FIG. 2B

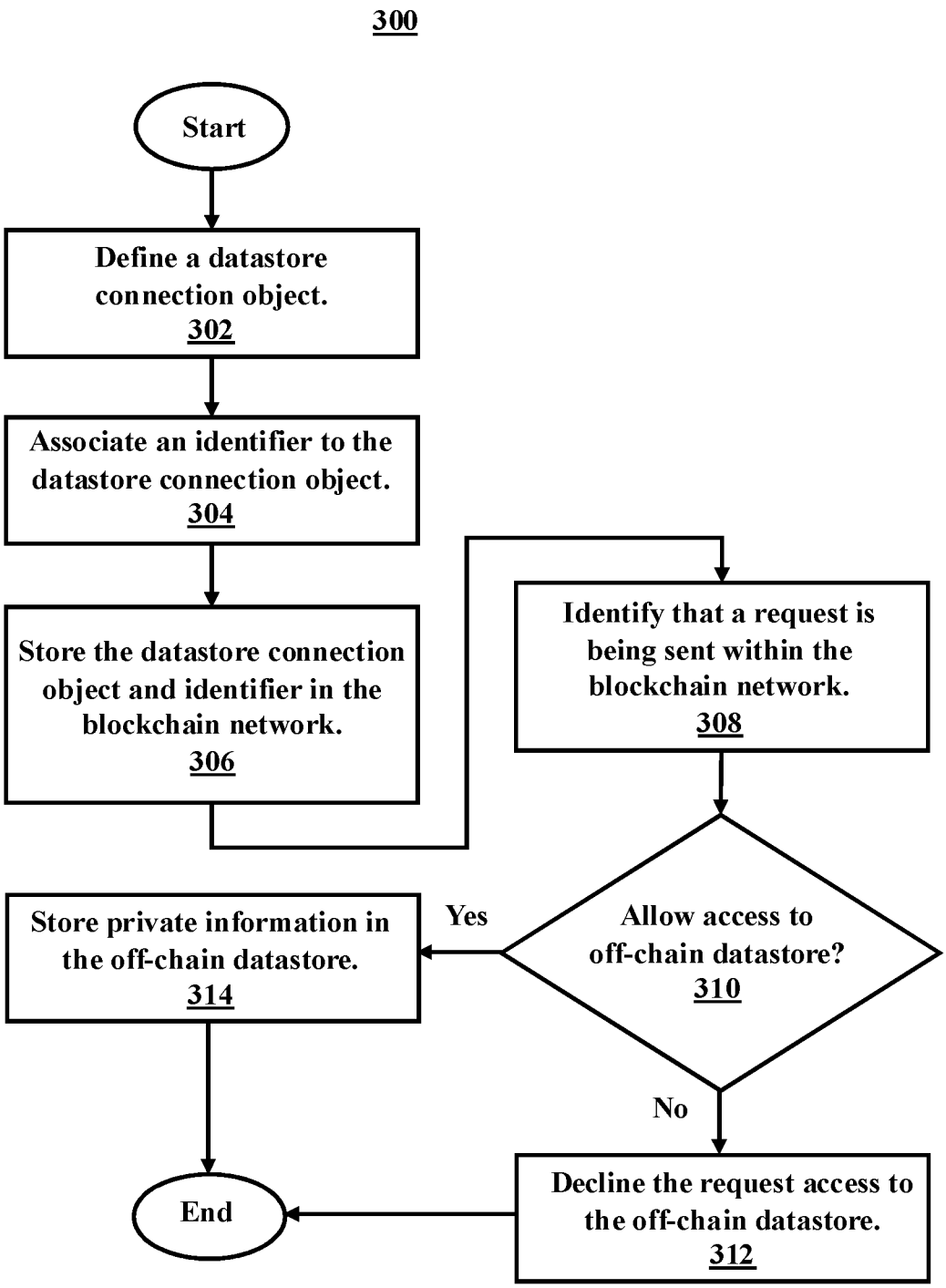
300
Start
Define a datastore connection object.
302
Associate an identifier to the datastore connection object.
304
Store the datastore connection object and identifier in the blockchain network.
306
Identify that a request is being sent within the blockchain network.
308
Allow access to off-chain datastore?
310
Yes
Store private information in the off-chain datastore.
314
No
Decline the request access to the off-chain datastore.
312
End

FIG. 3

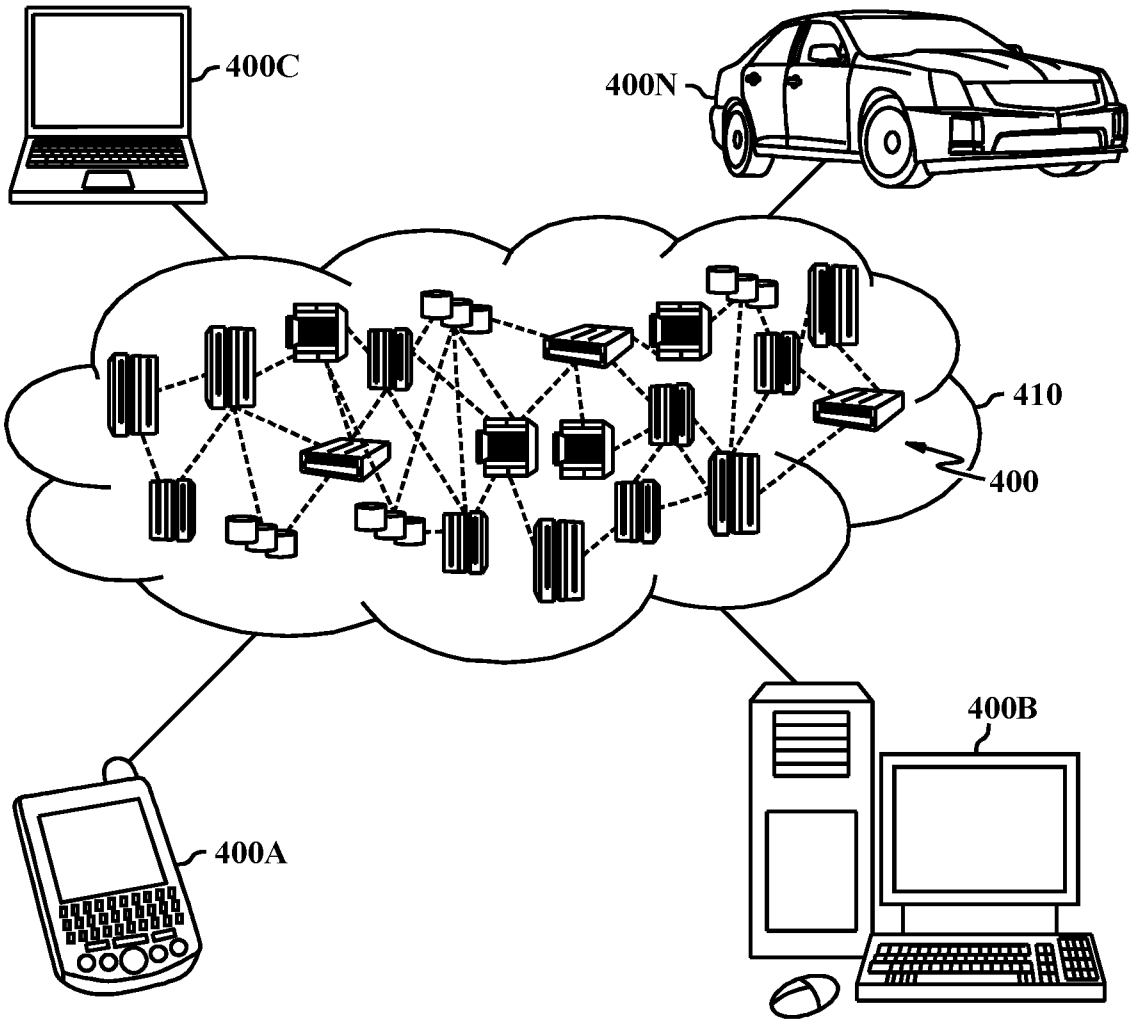

FIG. 4A

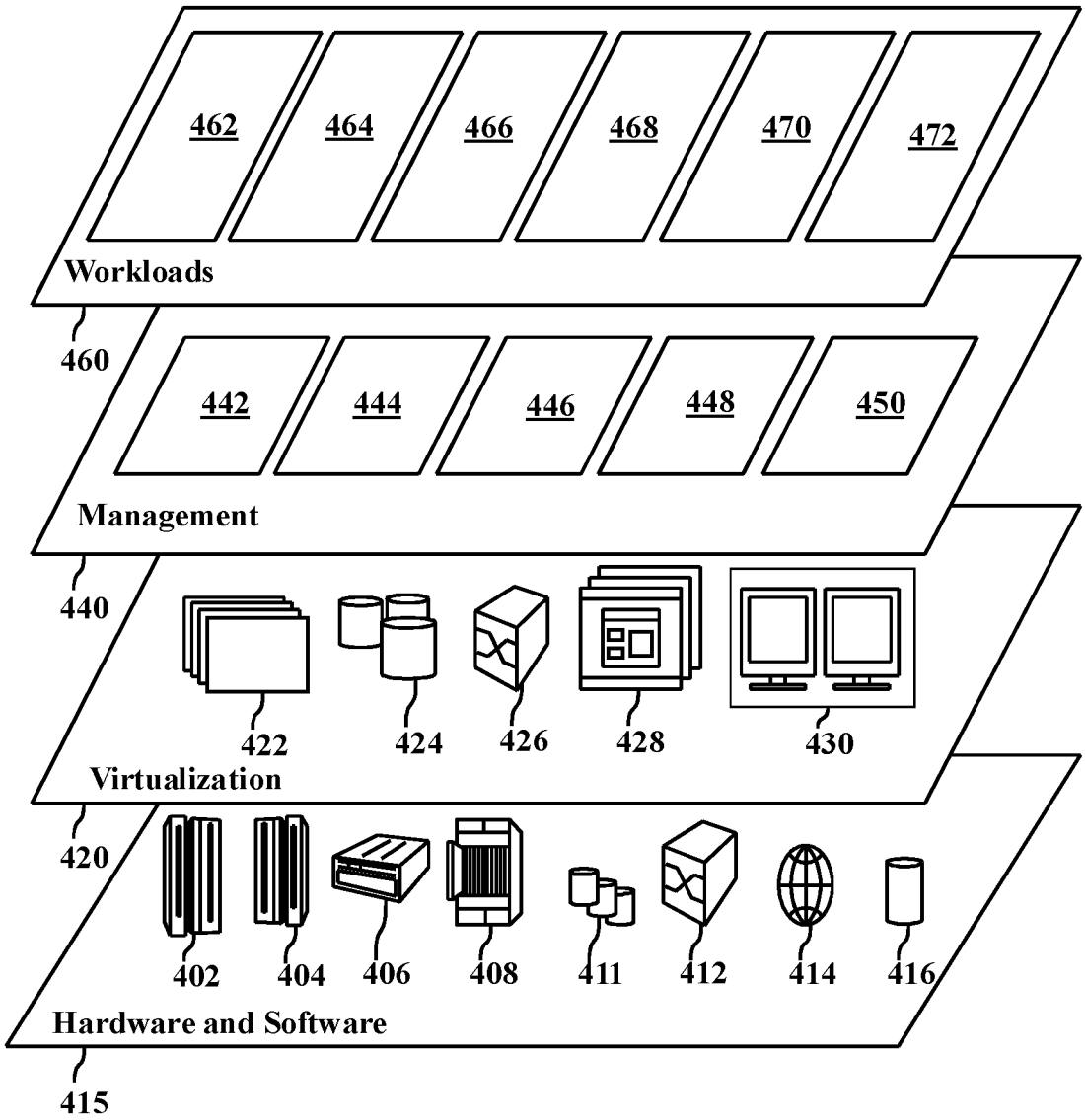
462
464
466
468
470
472
Workloads
460
442
444
446
448
450
Management
440
422
424
426
428
430
Virtualization
420
402
404
406
408
411
412
414
416
Hardware and Software
415

FIG. 4B

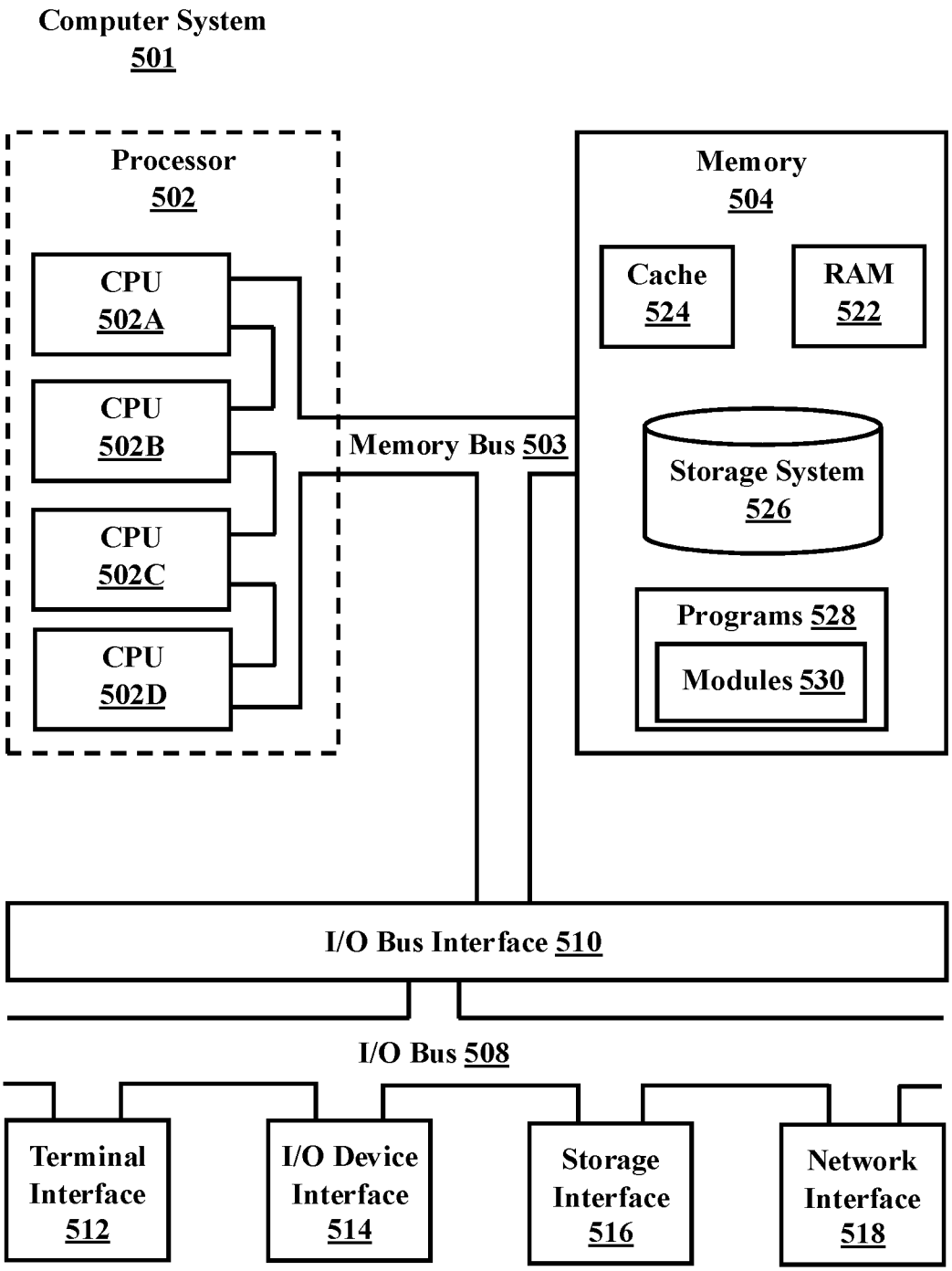
Computer System
501
Processor
502
CPU
502A
CPU
502B
CPU
502C
CPU
502D
Memory Bus 503
Memory
504
Cache
524
RAM
522
Storage System
526
Programs 528
Modules 530
I/O Bus Interface 510
I/O Bus 508
Terminal
Interface
512
I/O Device
Interface
514
Storage
Interface
516
Network
Interface
518

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2021/097738

A. CLASSIFICATION OF SUBJECT MATTER

G06F 21/62(2013.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WPI, EPODOC, CNPAT, CNKI: blockchain, block w chain, request, data, stor+, verif+, private, upload+, key

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	CN 109905474 A (SHANGHAI NANCHAO INFORMATION TECHNOLOGY CO., LTD.) 18 June 2019 (2019-06-18) paragraphs [0046]-[0112]	1-20
X	CN 110781508 A (SICHUAN CHANGHONG ELECTRIC CO., LTD.) 11 February 2020 (2020-02-11) paragraphs [0006]-[0025]	1-20
A	CN 111177277 A (ALIPAY HANGZHOU INFORMATION TECHNOLOGY CO., LTD.) 19 May 2020 (2020-05-19) the whole document	1-20
A	CN 110610101 A (BEIJING BAIDU NETCOM SCIENCE AND TECHNOLOGY CO., LTD.) 24 December 2019 (2019-12-24) the whole document	1-20

☐ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
13 August 2021	**30 August 2021**
Name and mailing address of the ISA/CN **National Intellectual Property Administration, PRC** **6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing** **100088** **China**	Authorized officer **JIAO,Yonghan**
Facsimile No. **(86-10)62019451**	Telephone No. **86-(10)-53961463**

Form PCT/ISA/210 (second sheet) (January 2015)

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2021/097738

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
CN	109905474	A	18 June 2019	None			
CN	110781508	A	11 February 2020	None			
CN	111177277	A	19 May 2020	CN	111177277	B	04 August 2020
				CN	112182099	A	05 January 2021
CN	110610101	A	24 December 2019	None			