

NOISY TRANSACTION FOR PROTECTION OF DATA

Background

[0001] A centralized database stores and maintains data in a single database (*e.g.*, a database server) at one location. This location is often a central computer, for example, a desktop central processing unit (CPU), a server CPU, or a mainframe computer. Information stored on a centralized database is typically accessible from multiple different points. Multiple users or client workstations can work simultaneously on the centralized database, for example, based on a client/server configuration. A centralized database is easy to manage, maintain, and control, especially for purposes of security because of its single location. Within a centralized database, data redundancy is minimized as a single storing place of all data also implies that a given set of data only has one primary record.

Summary

[0002] One example embodiment provides an apparatus that includes one or more of a processor configured to one or more of generate a blockchain request which comprises data for a blockchain, create a tag value which identifies whether the blockchain request is a noisy request that comprises fake data or a non-noisy request that comprises non-noisy data, and store the tag value within the blockchain request, and a network interface configured to transmit the blockchain request to one or more blockchain peers.

[0003] Another example embodiment provides a method that includes one or more of generating a blockchain request which comprises data for a blockchain, creating a tag value which identifies whether the blockchain request is a noisy request that comprises fake data or a non-noisy request that comprises non-noisy data, storing the tag value within the blockchain request, and transmitting the blockchain request to one or more blockchain peers.

[0004] A further example embodiment provides a non-transitory computer-readable medium comprising instructions, that when read by a processor, cause the processor to perform one or more of generating a blockchain request which comprises data for a blockchain, creating a tag value which identifies whether the blockchain request is a noisy request that comprises fake data or a non-
5 noisy request that comprises non-noisy data, storing the tag value within the blockchain request, and transmitting the blockchain request to one or more blockchain peers.

Brief Description of the Drawings

[0005] FIG. 1A is a diagram illustrating a computing environment for noisy transaction
10 submission according to example embodiments.

[0006] FIG. 1B is a diagram illustrating a process of recommending a noisy transaction according to example embodiments.

[0007] FIG. 2A is a diagram illustrating an example blockchain architecture configuration, according to example embodiments.

15 [0008] FIG. 2B is a diagram illustrating a blockchain transactional flow among nodes, according to example embodiments.

[0009] FIG. 3A is a diagram illustrating a permissioned network, according to example embodiments.

[0010] FIG. 3B is a diagram illustrating another permissioned network, according to example
20 embodiments.

[0011] FIG. 3C is a diagram illustrating a permissionless network, according to example embodiments.

[0012] FIG. 4A is a diagram illustrating a process of submitting a noisy transaction proposal to blockchain peers according to example embodiments.

[0013] FIG. 4B is a diagram illustrating content within a noisy transaction proposal according to example embodiments.

[0014] FIG. 5A is a diagram illustrating a method of processing a blockchain transaction based on a tag according to example embodiments.

5 [0015] FIG. 5B is a diagram illustrating a method storing a blockchain transaction including a tag according to example embodiments.

[0016] FIG. 6A is a diagram illustrating an example system configured to perform one or more operations described herein, according to example embodiments.

10 [0017] FIG. 6B is a diagram illustrating another example system configured to perform one or more operations described herein, according to example embodiments.

[0018] FIG. 6C is a diagram illustrating a further example system configured to utilize a smart contract, according to example embodiments.

[0019] FIG. 6D is a diagram illustrating yet another example system configured to utilize a blockchain, according to example embodiments.

15 [0020] FIG. 7A is a diagram illustrating a process of a new block being added to a distributed ledger, according to example embodiments.

[0021] FIG. 7B is a diagram illustrating data contents of a new data block, according to example embodiments.

20 [0022] FIG. 7C is a diagram illustrating a blockchain for digital content, according to example embodiments.

[0023] FIG. 7D is a diagram illustrating a block which may represent the structure of blocks in the blockchain, according to example embodiments.

[0024] FIG. 8A is a diagram illustrating an example blockchain which stores machine learning (artificial intelligence) data, according to example embodiments.

[0025] FIG. 8B is a diagram illustrating an example quantum-secure blockchain, according to example embodiments.

[0026] FIG. 9 is a diagram illustrating an example system that supports one or more of the example embodiments.

5

Detailed Description

[0027] 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. Thus, 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.

[0028] 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. Thus, 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 diagrams, 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.

25

[0029] 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.

5 [0030] Example embodiments provide methods, systems, components, non-transitory computer readable media, devices, and/or networks, which protect the confidentiality and the differential privacy of a blockchain ledger by purposely introducing noisy transactions (blockchain content) to a blockchain network.

[0031] In one embodiment the application utilizes a decentralized database (such as a blockchain)
10 that is a distributed storage system, which includes multiple nodes that communicate with each other. The decentralized database includes 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
15 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. In various embodiments, a permissioned and/or a permissionless blockchain can be used. In a public or permission-less blockchain, anyone can participate without a specific identity.
20 Public blockchains can involve native cryptocurrency and use consensus based on various protocols such as Proof of Work (PoW). 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, information, and the like.

[0032] Blockchain systems store data on an immutable ledger, provide distributed and
25 decentralized access to the immutable ledger through non-trusting participants, establish consensus

requirements for agreement between the non-trusting participants such that no one entity can change the immutable ledger without agreement from others, invoke smart contracts, and the like. A blockchain is formed by a network of participants which agree to add a block (with data stored therein) to the immutable ledger. Before being added, the block is linked to a previous block on the immutable ledger thereby forming a chain. This immutable and incorruptible nature of blockchain makes it safe from falsified information and hacks. The decentralized nature also gives it the unique quality of being trustless, in that parties do not need to establish trust before they can transact safely.

[0033] This application 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. The application 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 (also referred to herein as blockchain requests) associated with this application can be “endorsed” before being committed to the blockchain while transactions, which are not endorsed, are disregarded. 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.

[0034] This application 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). 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 transactions and modifying a world state of the blockchain, which is another name for the initial blockchain transaction which normally includes control and setup information.

[0035] This application 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 (i.e., 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.

[0036] This application can utilize a chain that is a transaction log which 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 (i.e., local, attached storage, cloud, etc.), efficiently supporting the append-only nature of the blockchain workload.

[0037] 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
5 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
10 before transactions are accepted.

[0038] A blockchain ledger is a trusted storage system because once the data has been added to the ledger, it is very difficult to change/tamper with. In a permissioned blockchain network, a transaction is considered valid if a minimum (usually required) number of endorser peers (*e.g.*, set forth by an endorsement policy) reach a consensus on the transaction by simulating the transaction
15 against a current state of the ledger and verifying a client signature added to the transaction. If the client signature is valid and the simulation is successful, the endorser node endorses (signs) the transaction and returns a successful endorsement response. A transaction proposal is then forwarded to an ordering service which adds the endorsed transaction to a block and broadcasts the block to the blockchain peers of the network for commitment.

20 [0039] Preserving data confidentiality in blockchain can be difficult. Any system that does not belong to a blockchain channel should not be able to interpret the ledger / state of the blockchain channel. However, attackers (*e.g.*, hackers, unauthorized systems, etc.) may attempt to steal a blockchain ledger by listening on an unauthorized channel associated with the blockchain. Accordingly, attackers from outside the blockchain can gain access to the content of the blockchain
25 ledger by sitting on the channel. Attacks may also come from inside the blockchain channel.

Members in the blockchain are usually required to share the ledger and the state. To protect this ledger data, members often encrypt the data which can incur significant overhead. Furthermore, sensitive data may be exposed based on transaction patterns of the members.

[0040] The benefits of the example embodiments include preventing attackers from interpreting the blockchain ledger and preserving data confidentiality by introducing noisy data (fake data) into the blockchain. For example, a client, a smart contract, an agent, etc., may introduce a noisy transaction which is processed like a typical blockchain transaction but with a unique identifier that indicates the transaction is noisy. Using the unique identifier, the noisy transaction data can be identified and ignored by member/participants of the blockchain which are involved in the noisy transaction. However, other blockchain users and unauthorized users that are not aware of the unique identifier may process the noisy transaction as usual and incorrectly update their copy of the blockchain ledger thereby creating an obscured state which protects the confidentiality of the clients involved in the noisy transaction. As a result, the unauthorized users are not be able to correctly interpret the ledger when it comes to the transaction history of such clients. Accordingly, the state of the blockchain ledger can be preserved without introducing significant performance and management overhead among the blockchain peers.

[0041] A noisy transaction may only be understood by a creator of the noisy transaction (*e.g.*, client, peer, etc.) and any accessors / participants of the blockchain that are made aware of the noisy transaction. Clients and/or blockchain peers may collaboratively manage and share keys which can be used to encrypt/decrypt a tag. The tag may be a value that specifies whether a transaction is noisy or real data (non-noise). For example, the tag may include a Boolean value with one value indicating noisy and the other value indicating non-noisy, however, embodiments are not limited thereto. The tag may encrypted such that it is only understandable to the clients / blockchain peers that are part of the noisy transaction and that have the decryption key. Here, the tag may be encrypted with a public key and the participating clients and peers may share the corresponding

private key capable of successfully decrypting the tag and reading the tag to understand whether the transaction is noise or not.

[0042] The key may be an encryption key that is shared among the participants. For example, the participants may implement a key sharing scheme such as decentralized public key infrastructure (PKI), peer-to-peer (P2P), and the like. Peers and clients that are made aware of the key may detect that the noisy transaction is noisy by decrypting the tag value. However, peers and clients that do not receive the key cannot decrypt the tag to tell if the transaction is noisy, and will treat the transaction as normal. Thus, the unaware peer or client will update the blockchain and the state database incorrectly. In some embodiments, the blockchain network may use multiple keys to encryption tag values. In this case, the client and/or blockchain peer that creates the transaction may also add a key identifier to the transaction to identify which shared secret key is used to encrypt the tag value.

[0043] A noisy transaction (also referred to herein as a noisy blockchain request) may be created collaboratively by a group of the clients that have data in the world state database that will be affected by the noisy transaction. In this case, the noisy transaction will be understood as noise by the group of clients, but not be understood as noise by other unrelated clients. Therefore, when unrelated clients look back at the noisy transaction (*e.g.*, in the future), the noisy transaction will appear as a normal/typical transaction. In generally, the noisy transaction will only be understood as noise by the subset of the clients that created the noisy transaction. In doing so, clients can disguise transaction patterns including frequency of transaction, content of transaction, and the like, to preserve their data privacy on the blockchain. Furthermore, every transaction (whether noisy or not noisy) may include a tag value which is obscured and only capable of being revealed by clients with access to shared secret.

[0044] For example, a noisy transaction may add additional shipping records and dates in order to hide a shipping pattern. In this example, new key-values reflecting these additional noisy shipping

records will be added by uninformed peers to the blockchain ledger (world state database) and viewed by uninformed clients. As another example, a noisy transaction may add additional sales activity between a buyer and a seller in order to obscure a contractual business relationship. In this case, the additional sales activity will be added by uninformed peers to the blockchain ledger (world state database). There are many other scenarios possible, and these examples are not meant to be limiting.

[0045] The use of noisy transactions may protect the privacy of the transactions themselves because an attacker does not know whether a specific transaction is valid data or noisy data. As another example, the use of noisy transactions may protect user privacy so that an attacker cannot tell whether a transaction pattern belongs to a particular user. In some embodiments, noisy transactions may be created randomly. As another example, noisy transactions may be created based on historical transaction data stored on the blockchain. In other words, a smart contract, agent, client, etc., may monitor and identify patterns of behavior on the blockchain and introduce noisy transactions based on such patterns of behavior. The pattern of behavior may be used to identify a “believable” noisy transaction. As another example, the pattern of behavior may be used to create a new pattern of behavior with the noisy transaction.

[0046] The example embodiments provide a lightweight approach to protect the content of the blockchain using differential privacy. Noisy transactions and users may be injected into the blockchain system to prevent an attacker from learning privacy related knowledge from the blockchain. A unique tag may be used by member participant peers to interpret whether a transaction is noisy or real/live data. Furthermore, the noisy information within the noisy transaction may be organized in a way that it does not affect the correctness of normal blockchain operations (e.g., endorsement, consensus, read-write sets, etc.).

[0047] FIG. 1A illustrates a computing environment 100A for noisy transaction submission according to example embodiments. Referring to FIG. 1A, a client 121 and a client 122 conduct

transactions with one another (*e.g.*, transfer funds, Internet of Things (IoT) data, etc.). The transactions may be processed by a network of blockchain peers 111, 112, 113, 114, and 115. For example, transaction data may be endorsed by endorser nodes among the blockchain peers, added/ordered within a block, and recorded on a blockchain 110. In a typical blockchain transaction

5 (*i.e.*, simply referred to herein as a transaction), the transaction data is assumed to be live data. In the example embodiments, however, a transaction may be a noisy transaction that includes fake data.

[0048] For example, a smart contract, a client, an agent, etc., may randomly generate a noisy transaction between client 121 and client 122. The randomness may have a differential privacy guarantee. As another example, (shown in FIG. 1B), the smart contract, the client, the agent, etc.,

10 may analyze historical ledger data and recommend a noisy transaction. Therefore, a noisy transaction proposal may be submitted by the client 121, the client 122, a blockchain peer, or the like, and processed by the blockchain network of peers shown in FIG. 1A. In this example, all of the blockchain peers except blockchain peer 113 are aware of an encryption key that is used to obscure a noisy transaction. For example, a tag value may be added to a transaction proposal and it

15 may be encrypted using the encryption key. The tag value is a piece of encrypted information in a blockchain transaction. The same tag may have different meanings to different blockchain clients, depending on whether a client can decrypt this tag or not. If a client can decrypt the tag, the client can tell whether this transaction is a noisy one or not based on the decrypted tag value. If a client cannot decrypt the tag, the client has to treat this transaction as a non-noisy transaction.

20 **[0049]** In this case, the transaction data may be processed normally by the blockchain network (endorsement, consensus, etc.) and added to a block that is submitted to each of the blockchain peers 111-115 for storage. However, the transaction that is noisy is not to be stored/recorded within the world state database on the blockchain ledger and it is not to affect the blockchain state. Blockchain peers 111, 112, 114, and 115 may detect the noisy transaction and ignore it by not updating the

25 world state database when updating/committing the new block to the blockchain ledger. However,

blockchain peer 113 is not able to detect the noisy transaction since it cannot decrypt the tag value. Therefore, the blockchain peer 113 will process the noisy transaction as normal and incorrectly update its local copy of the worldstate database on the blockchain ledger thereby obscuring the state of the blockchain for the transactions involving the clients of the noisy transaction.

5 **[0050]** FIG. 1B illustrates a process 100B of recommending a noisy transaction according to example embodiments. Referring to FIG. 1B, a smart contract 140 may analyze historical transaction data stored on the blockchain ledger 110 and recommend a noisy transaction 142 based on the historical transaction data on the blockchain ledger 110. It should be appreciated that other entities may recommend noisy transactions instead of the smart contract 140 including, but not
10 limited to, clients, agents, and the like.

[0051] In this example, the smart contract 140 may analyze historical client data 131, 132, 133, and the like. Here, the smart contract 140 may analyze the historical client data 131, 132, and 133, and recommend a noisy transaction between client A and client B. From the historical data 131, 132, and 133, the smart contract 140 may derive the frequencies of a client with every other client in the
15 system in a given period. For example, a frequency distribution may be identified from the historical data 131, 132, and 133. The frequency distribution may include a transaction pattern of the clients A and B. The smart contract 140 may decide how to perturb this frequency distribution with random noise with differential privacy guarantee. The perturbation of frequency distribution may be achieved by adding noisy transactions with other clients in the system, or adding fake
20 transactions between clients A and B. As further described with respect to FIGS. 4A-4B, a unique tag may be added to all transactions to identify / distinguish noisy transactions from live / real transactions.

[0052] After determining a suggested / recommended noisy transaction 142, the smart contract 140 and/or the blockchain peer may provide the suggested noisy transaction 142 to one or more of
25 the clients 121 and 122 shown in FIG. 1A. Here, both the clients 121 and 122 may approve of a

noisy transaction involving them and even submit the noisy transaction into the blockchain by submitting the transaction to one or more endorser peers. Although not shown in FIG. 1A, the client that submits the noisy transaction to the blockchain may be a noisy client in that they are not actively participating in the blockchain but that can be identified as an actual entity.

- 5 **[0053]** FIG. 2A illustrates a blockchain architecture configuration 200, according to example embodiments. Referring to FIG. 2A, the blockchain architecture 200 may include certain blockchain elements, for example, a group of blockchain nodes 202. The blockchain nodes 202 may include one or more nodes 204-210 (these four nodes are depicted by example only). These nodes participate in a number of activities, such as blockchain transaction addition and validation process (consensus).
- 10 One or more of the blockchain nodes 204-210 may endorse transactions based on endorsement policy and may provide an ordering service for all blockchain nodes in the architecture 200. A blockchain node may initiate a blockchain authentication and seek to write to a blockchain immutable ledger stored in blockchain layer 216, a copy of which may also be stored on the underpinning physical infrastructure 214. The blockchain configuration may include one or more
- 15 applications 224 which are linked to application programming interfaces (APIs) 222 to access and execute stored program/application code 220 (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 204-210.
- 20 **[0054]** The blockchain base or platform 212 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 216 may expose an interface that provides access to the virtual execution environment necessary to process the program

code and engage the physical infrastructure 214. Cryptographic trust services 218 may be used to verify transactions such as asset exchange transactions and keep information private.

[0055] The blockchain architecture configuration of FIG. 2A may process and execute program/application code 220 via one or more interfaces exposed, and services provided, by blockchain platform 212. The code 220 may control blockchain assets. For example, the code 220 can store and transfer data, and may be executed by nodes 204-210 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 reminders, updates, and/or other notifications subject to the changes, updates, etc. The smart contracts can themselves be used to identify rules associated with authorization and access requirements and usage of the ledger. For example, the smart contract (or chaincode executing the logic of the smart contract) may read blockchain data 226 which may be processed by one or more processing entities (e.g., virtual machines) included in the blockchain layer 216 to generate results 228 including alerts, determining liability, and the like, within a complex service scenario. The physical infrastructure 214 may be utilized to retrieve any of the data or information described herein.

[0056] 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., distributed network of blockchain peers). A transaction is an execution of the smart contract code which 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.

[0057] 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 one or more blocks within 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.

[0058] A chaincode may include the code interpretation (*e.g.*, the logic) of a smart contract. For example, the chaincode may include a packaged and deployable version of the logic within the smart contract. 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 may receive a hash and retrieve 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.

[0059] FIG. 2B illustrates an example of a blockchain transactional flow 250 between nodes of the blockchain in accordance with an example embodiment. Referring to FIG. 2B, the transaction flow may include a client node 260 transmitting a transaction proposal 291 to an endorsing peer node 281. The endorsing peer 281 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). Here, the endorsing peer 281 may determine whether or not to endorse the transaction proposal. The proposal response 292 is sent back to the client 260 along with an endorsement signature, if approved. The client 260 assembles the endorsements into a transaction payload 293 and broadcasts it to an ordering service node 284. The ordering service node 284 then delivers ordered transactions

as blocks to all peers 281-283 on a channel. Before committal to the blockchain, each peer 281-283 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 293.

5 **[0060]** Furthermore, prior to committing each transaction to the blockchain, the peers 281-283 may check a tag value (which may be encrypted) that is included within the transaction to determine whether the transaction is an actual / real transaction or a noisy transaction with fake data. If the tag indicates the transaction is noisy, the peers 281-283 may determine the transaction is fake and ignore the transaction. However, if the tag indicates the transaction is non-noisy (*i.e.*, actual data to
10 be stored on the blockchain), the peers 281-283 may store the data within their local copy of the worldstate database.

[0061] Referring again to FIG. 2B, the client node initiates the transaction 291 by constructing and sending a request to the peer node 281, which is an endorser. The client 260 may include an application leveraging a supported software development kit (SDK), which utilizes an available API
15 to generate a transaction proposal. The proposal is a request to invoke a chaincode function so that data can be read and/or written to the ledger (*i.e.*, write new key value pairs for the assets). The SDK may serve as a shim to package the transaction proposal 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.

20 **[0062]** In response, the endorsing peer node 281 may verify (a) that the transaction proposal 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 260, in the example) is properly authorized to perform the proposed operation on that channel. The endorsing peer node 281 may take the transaction proposal inputs as arguments to the invoked chaincode function. The chaincode
25 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 292, the set of values, along with the endorsing peer node's 281 signature is passed back as a proposal response 292 to the SDK of the client 260 which parses the payload for the application to consume.

[0063] In response, the application of the client 260 inspects/verifies the endorsing peers
5 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 284. If the client application intends to submit the transaction to the ordering node service 284 to update the ledger, the application determines if the specified endorsement policy has been fulfilled before submitting (i.e., did all peer
10 nodes necessary for the transaction endorse the transaction). 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.

15 [0064] After successful inspection, in step 293 the client 260 assembles endorsements into a transaction proposal and broadcasts the transaction proposal and response within a transaction message to the ordering node 284. The transaction may contain the read/write sets, the endorsing peers signatures and a channel ID. The ordering node 284 does not need to inspect the entire content of a transaction in order to perform its operation, instead the ordering node 284 may simply receive
20 transactions from all channels in the network, order them chronologically by channel, and create blocks of transactions per channel.

[0065] The blocks are delivered from the ordering node 284 to all peer nodes 281-283 on the channel. The data section within the block may be validated to ensure an endorsement policy is fulfilled and to ensure that there have been no changes to ledger state for read set variables since the
25 read set was generated by the transaction execution. In addition, in 294, a tag value may be

decrypted and checked to ensure that transaction data is not noisy transaction data. Furthermore, in step 295 each peer node 281-283 appends the block to the channel's chain, and for each valid transaction the write sets are committed to current state database as long as the transaction is not a noisy transaction. An event may be emitted, to notify the client application that the transaction
5 (invocation) has been immutably appended to the chain, as well as to notify whether the transaction was validated or invalidated.

[0066] FIG. 3A illustrates an example of a permissioned blockchain network 300, which features a distributed, decentralized peer-to-peer architecture. In this example, a blockchain user 302 may initiate a transaction to the permissioned blockchain 304. In this example, the transaction can be a
10 deploy, invoke, or query, and may be issued through a client-side application leveraging an SDK, directly through an API, etc. Networks may provide access to a regulator 306, such as an auditor. A blockchain network operator 308 manages member permissions, such as enrolling the regulator 306 as an "auditor" and the blockchain user 302 as a "client". An auditor could be restricted only to querying the ledger whereas a client could be authorized to deploy, invoke, and query certain types
15 of chaincode.

[0067] A blockchain developer 310 can write chaincode and client-side applications. The blockchain developer 310 can deploy chaincode directly to the network through an interface. To include credentials from a traditional data source 312 in chaincode, the developer 310 could use an out-of-band connection to access the data. In this example, the blockchain user 302 connects to the
20 permissioned blockchain 304 through a peer node 314. Before proceeding with any transactions, the peer node 314 retrieves the user's enrollment and transaction certificates from a certificate authority 316, which manages user roles and permissions. In some cases, blockchain users must possess these digital certificates in order to transact on the permissioned blockchain 304. Meanwhile, a user attempting to utilize chaincode may be required to verify their credentials on the traditional data

source 312. To confirm the user's authorization, chaincode can use an out-of-band connection to this data through a traditional processing platform 318.

[0068] FIG. 3B illustrates another example of a permissioned blockchain network 320, which features a distributed, decentralized peer-to-peer architecture. In this example, a blockchain user 322 may submit a transaction to the permissioned blockchain 324. In this example, the transaction can be a deploy, invoke, or query, and may be issued through a client-side application leveraging an SDK, directly through an API, etc. Networks may provide access to a regulator 326, such as an auditor. A blockchain network operator 328 manages member permissions, such as enrolling the regulator 326 as an "auditor" and the blockchain user 322 as a "client". An auditor could be restricted only to querying the ledger whereas a client could be authorized to deploy, invoke, and query certain types of chaincode.

[0069] A blockchain developer 330 writes chaincode and client-side applications. The blockchain developer 330 can deploy chaincode directly to the network through an interface. To include credentials from a traditional data source 332 in chaincode, the developer 330 could use an out-of-band connection to access the data. In this example, the blockchain user 322 connects to the network through a peer node 334. Before proceeding with any transactions, the peer node 334 retrieves the user's enrollment and transaction certificates from the certificate authority 336. In some cases, blockchain users must possess these digital certificates in order to transact on the permissioned blockchain 324. Meanwhile, a user attempting to utilize chaincode may be required to verify their credentials on the traditional data source 332. To confirm the user's authorization, chaincode can use an out-of-band connection to this data through a traditional processing platform 338.

[0070] In some embodiments, the blockchain herein may be a permissionless blockchain. In contrast with permissioned blockchains which require permission to join, anyone can join a permissionless blockchain. For example, to join a permissionless blockchain a user may create a personal address and begin interacting with the network, by submitting transactions, and hence

adding entries to the ledger. Additionally, all parties have the choice of running a node on the system and employing the mining protocols to help verify transactions.

[0071] FIG. 3C illustrates a process 350 of a transaction being processed by a permissionless blockchain 352 including a plurality of nodes 354. A sender 356 desires to send payment or some other form of value (*e.g.*, a deed, medical records, a contract, a good, a service, or any other asset that can be encapsulated in a digital record) to a recipient 358 via the permissionless blockchain 352. In one embodiment, each of the sender device 356 and the recipient device 358 may have digital wallets (associated with the blockchain 352) that provide user interface controls and a display of transaction parameters. In response, the transaction is broadcast throughout the blockchain 352 to the nodes 354. Depending on the blockchain's 352 network parameters the nodes verify 360 the transaction based on rules (which may be pre-defined or dynamically allocated) established by the permissionless blockchain 352 creators. For example, this may include verifying identities of the parties involved, etc. The transaction may be verified immediately or it may be placed in a queue with other transactions and the nodes 354 determine if the transactions are valid based on a set of network rules.

[0072] In structure 362, valid transactions are formed into a block and sealed with a lock (hash). This process may be performed by mining nodes among the nodes 354. Mining nodes may utilize additional software specifically for mining and creating blocks for the permissionless blockchain 352. Each block may be identified by a hash (*e.g.*, 256 bit number, etc.) created using an algorithm agreed upon by the network. Each block may include a header, a pointer or reference to a hash of a previous block's header in the chain, and a group of valid transactions. The reference to the previous block's hash is associated with the creation of the secure independent chain of blocks.

[0073] Before blocks can be added to the blockchain, the blocks must be validated. Validation for the permissionless blockchain 352 may include a proof-of-work (PoW) which is a solution to a puzzle derived from the block's header. Although not shown in the example of FIG. 3C, another

process for validating a block is proof-of-stake. Unlike the proof-of-work, where the algorithm rewards miners who solve mathematical problems, with the proof of stake, a creator of a new block is chosen in a deterministic way, depending on its wealth, also defined as “stake.” Then, a similar proof is performed by the selected/chosen node.

5 **[0074]** With mining 364, nodes try to solve the block by making incremental changes to one variable until the solution satisfies a network-wide target. This creates the PoW thereby ensuring correct answers. In other words, a potential solution must prove that computing resources were drained in solving the problem. In some types of permissionless blockchains, miners may be rewarded with value (*e.g.*, coins, etc.) for correctly mining a block.

10 **[0075]** Here, the PoW process, alongside the chaining of blocks, makes modifications of the blockchain extremely difficult, as an attacker must modify all subsequent blocks in order for the modifications of one block to be accepted. Furthermore, as new blocks are mined, the difficulty of modifying a block increases, and the number of subsequent blocks increases. With distribution 366, the successfully validated block is distributed through the permissionless blockchain 352 and all
15 nodes 354 add the block to a majority chain which is the permissionless blockchain’s 352 auditable ledger. Furthermore, the value in the transaction submitted by the sender 356 is deposited or otherwise transferred to the digital wallet of the recipient device 358.

20 **[0076]** FIG. 4A illustrates a process 400A of submitting a noisy transaction proposal 440 to blockchain peers according to example embodiments, and FIG. 4B illustrates the content within a transaction proposal 440 according to example embodiments. Referring to FIG. 4A, a noisy transaction may be transmitted from a client 402 to one or more endorser nodes (not shown) as described in the example of FIG. 2B. If properly endorsed, the client 402 may submit the transaction proposal 440 based on the endorsements to an ordering service (*e.g.*, orderer node 410) of the blockchain network. In response, the orderer node 410 may combine the transaction proposal
25 440 with other transaction proposals which may include non-noisy transaction proposals and/or

noisy transaction proposals and create a new block 450 to be added to a distributed blockchain ledger of the blockchain network. Once the block 450 is created, the orderer node 410 may broadcast the block 450 to the blockchain peers of the blockchain network for validation and commitment to their respective blockchain ledgers.

5 **[0077]** In the example of FIG. 4A, two blockchain peers 420 and 430 are shown each storing their own version of the blockchain ledger (blockchain ledgers 422 and blockchain ledger 432). In this example, blockchain peer 420 is an authorized participant of the blockchain network and includes a shared secret key that is shared by the clients and/or blockchain peers that submit the noisy data. Meanwhile, blockchain peer 430 is an unauthorized peer that is listening on a channel of the

10 blockchain network. Here, the blockchain peer 430 does not have a corresponding shared secret key.

[0078] Each blockchain peer 420 and 430 may receive the block 450 and update their respective blockchain ledgers 422 and 432 based on the individual transactions within the block 450 including the transaction proposal 440. The content within the transaction proposal 440 may include a tag value 446 which identifies whether the transaction proposal 440 is a noisy transaction (fake data) or
15 a real transaction. In this case, the tag value 446 indicates the transaction is fake/noisy. It should also be appreciated that even non-noisy transactions that include valid / live data may also include a tag value which identifies the transaction as non-noisy (valid, live, etc.).

[0079] The tag value 446 may be encrypted with a shared encryption key that is known to blockchain peer 420 but not known to the unauthorized peer 430. The blockchain peer 420 may
20 detect that the noisy transaction proposal 440 is a noisy transaction by decrypting the tag value 446 and identifying a Boolean indicator (or some other tag value). Thus, the blockchain peer 420 may skip or otherwise ignore the state changes to the blockchain ledger within the noisy transaction proposal 440. Meanwhile, the unauthorized peer 430 is not able to decrypt such tag value 446 and is unable to determine that the transaction proposal 440 is noise. Therefore, the unauthorized peer 430

will process the noisy transaction proposal 440 as a typical transaction and update the blockchain ledger 432 accordingly.

[0080] In FIG. 4A, blockchain peer 420 has its own copy of the blockchain ledger (ledger 422) which is correct (*i.e.*, unobscured state) because the noisy transaction proposal 440 is ignored and not added to a worldstate database on the blockchain ledger 422. Meanwhile, the blockchain ledger 432 of the blockchain peer 430 is in an obscured state since the content from the noisy transaction proposal 440 has been added to its ledger 432 (*e.g.*, updated key-value pairs in a state database 460). Thus, the transaction patterns of the clients involved in the noisy transaction is obscured and protected.

[0081] FIG. 4B illustrates an example of the content of a transaction proposal 440 described with respect to FIG. 4A. The transaction proposal 440 may be stored in the block 450 along with other transaction proposals. When processed, the transaction proposals may be used to update key-value pairs stored within the world state 460 which stores a most recent value of all key-value pairs on the blockchain. As shown in FIG. 4B, the transaction proposal 440 includes the same components as a normal blockchain transaction with the difference being the tag value 446 which may be encrypted and which identifies whether the transaction is noisy or not, and a key identifier 447 which identifies an encryption key that is used to encrypt the tag value 446. By keeping the transaction format the same, an unauthorized party is unaware that noisy data is possible.

[0082] For example, the transaction proposal 440 may include a header 441 that captures essential metadata about the transaction such as the name of the chaincode and its version, and a signature 442 that contains a cryptographic signature of the client 402 that created the submitted the transaction proposal 440. The transaction proposal 440 also includes endorsements 445 which include a list of signed transaction responses from each required endorser to satisfy an endorsement policy.

[0083] The transaction proposal 440 also includes a proposal 443 with input parameters for a smart contract, and a response 444 which include read-write sets of the endorser nodes. In this example, the transaction proposal 440 is a noisy transaction. Therefore, the proposal 443 includes noisy / fake data. The proposal 443 may encode noisy input parameters supplied to a smart contract / chaincode which creates the proposed noisy update to the blockchain ledger. When the logic of a smart contract runs, the proposal 443 provides the noisy input parameters (*e.g.*, key-value pairs, etc.) which are used to determine the new world state with respect to the current world state. The transaction proposal 440 also includes a fake response 444 which captures the before and after values of the world state as read-write sets with noisy data. The response 444 is the output of the smart contract based on the noisy input from the proposal 443.

[0084] FIG. 5A illustrates a method 500 of processing a blockchain transaction based on a tag according to example embodiments. As a non-limiting example, the method 500 may be performed by a client application, a smart contract on a blockchain peer, an agent system of the blockchain, and the like. Referring to FIG. 5A, in 502, the method may include generating a blockchain request which comprises data for a blockchain. For example, the blockchain request may include a blockchain transaction that is configured to invoke chaincode. The blockchain request may include key-values to be modified on the blockchain ledger.

[0085] In some embodiments, the generating may include generating a noisy blockchain request that comprises fake data for the blockchain, and the creating comprises creating a tag value which identifies the blockchain request as noisy. In this example, the fake data may include fake key-values for the blockchain, and the tag value comprises a Boolean value. In some embodiments, the generating may include generating a non-noisy blockchain request that comprises live data for the blockchain, and the creating comprises creating a tag value which identifies that the blockchain request as live data.

[0086] In 504, the method may include creating a tag value which identifies whether the blockchain request is a noisy request that comprises fake data or a non-noisy request that comprises non-noisy data. For example, the tag value may include a Boolean value where “true” indicates noisy and “false” indicates non-noisy, or vice versa. In some embodiments, the method may further include obscuring the tag value based on a secret key, and storing a key identifier value within the blockchain request which identifies the secret key used to obscure the tag value. For example, the key identifier may differentiate which encryption key is used when multiple encryption keys are used within the blockchain network. In some embodiments, the secret key may include a public key of a symmetric public and private key pair, and the private key is shared by blockchain peers that are members of the blockchain.

[0087] In 506, the method may include storing the tag value within the blockchain request, and in 508, the method may include transmitting the blockchain request to one or more blockchain peers. In some embodiments, the method may further include receiving signatures and read-write sets with respect to the noisy blockchain request from endorser peers of the blockchain, and storing the signatures and the read-write sets in the noisy blockchain request. In some embodiments, the generating may include generating a noisy blockchain request based on a historical pattern of behavior of a client on the blockchain.

[0088] FIG. 5B illustrates a method 510 of storing a blockchain transaction including a tag according to example embodiments. Referring to FIG. 5B, in 512, the method may include receiving a blockchain request that comprises a tag value which identifies whether the blockchain request is a noisy request that comprises fake data or a non-noisy request that comprises non-noisy data. For example, the blockchain request may be a noisy blockchain request that comprises fake data for a blockchain, and the tag value may include a tag value which identifies the blockchain request as noisy. In some embodiments, the fake data may include fake key-values for the

blockchain, and the tag value may include a hash value previously agreed to by creators of the noisy blockchain request.

[0089] In 514, the method may include verifying that the blockchain request has been endorsed by enough blockchain peers to satisfy a predetermined policy. For example, the verifying may include
5 verifying signatures and read-write sets with respect to the noisy blockchain request from endorser peers of the noisy blockchain request. In response to successful verification, in 516, the method may include storing the blockchain request which includes the tag value within a data block among a hash-linked chain of data blocks.

[0090] In some embodiments, the tag value may include an obscured tag value that has been
10 obscured based on a secret key, and the processor is further configured to store a key identifier value of the secret key within the data block. In some embodiments, the secret key may include a public key of a symmetric public and private key pair, and the private key is shared by blockchain clients within a blockchain network that created the noisy blockchain request but not shared by other blockchain clients within the blockchain network. In some embodiments, the blockchain request
15 may be a non-noisy blockchain request that comprises live data for the blockchain, and the tag value may include a value which identifies that the blockchain request is not noise to blockchain clients that created the blockchain request.

[0091] FIG. 6A illustrates an example system 600 that includes a physical infrastructure 610 configured to perform various operations according to example embodiments. Referring to FIG. 6A,
20 the physical infrastructure 610 includes a module 612 and a module 614. The module 614 includes a blockchain 620 and a smart contract 630 (which may reside on the blockchain 620), that may execute any of the operational steps 608 (in module 612) included in any of the example embodiments. The steps/operations 608 may include one or more of the embodiments described or depicted and may represent output or written information that is written or read from one or more
25 smart contracts 630 and/or blockchains 620. The physical infrastructure 610, the module 612, and

the module 614 may include one or more computers, servers, processors, memories, and/or wireless communication devices. Further, the module 612 and the module 614 may be a same module.

[0092] FIG. 6B illustrates another example system 640 configured to perform various operations according to example embodiments. Referring to FIG. 6B, the system 640 includes a module 612
5 and a module 614. The module 614 includes a blockchain 620 and a smart contract 630 (which may reside on the blockchain 620), that may execute any of the operational steps 608 (in module 612) included in any of the example embodiments. The steps/operations 608 may include one or more of the embodiments described or depicted and may represent output or written information that is written or read from one or more smart contracts 630 and/or blockchains 620. The physical
10 infrastructure 610, the module 612, and the module 614 may include one or more computers, servers, processors, memories, and/or wireless communication devices. Further, the module 612 and the module 614 may be a same module.

[0093] FIG. 6C illustrates an example system configured to utilize a smart contract configuration among contracting parties and a mediating server configured to enforce the smart contract terms on
15 the blockchain according to example embodiments. Referring to FIG. 6C, the configuration 650 may represent a communication session, an asset transfer session or a process or procedure that is driven by a smart contract 630 which explicitly identifies one or more user devices 652 and/or 656. The execution, operations and results of the smart contract execution may be managed by a server 654. Content of the smart contract 630 may require digital signatures by one or more of the entities 652
20 and 656 which are parties to the smart contract transaction. The results of the smart contract execution may be written to a blockchain 620 as a blockchain transaction. The smart contract 630 resides on the blockchain 620 which may reside on one or more computers, servers, processors, memories, and/or wireless communication devices.

[0094] FIG. 6D illustrates a system 660 including a blockchain, according to example
25 embodiments. Referring to the example of FIG. 6D, an application programming interface (API)

gateway 662 provides a common interface for accessing blockchain logic (*e.g.*, smart contract 630 or other chaincode) and data (*e.g.*, distributed ledger, etc.). In this example, the API gateway 662 is a common interface for performing transactions (invoke, queries, etc.) on the blockchain by connecting one or more entities 652 and 656 to a blockchain peer (*i.e.*, server 654). Here, the server 5 654 is a blockchain network peer component that holds a copy of the world state and a distributed ledger allowing clients 652 and 656 to query data on the world state as well as submit transactions into the blockchain network where, depending on the smart contract 630 and endorsement policy, endorsing peers will run the smart contracts 630.

[0095] The above embodiments may be implemented in hardware, in a computer program 10 executed by a processor, in firmware, or in a combination of the above. A computer program may be embodied on a computer readable medium, such as a storage medium. For example, a computer program may reside in random access memory (“RAM”), flash memory, read-only memory (“ROM”), erasable programmable read-only memory (“EPROM”), electrically erasable programmable read-only memory (“EEPROM”), registers, hard disk, a removable disk, a compact 15 disk read-only memory (“CD-ROM”), or any other form of storage medium known in the art.

[0096] An exemplary storage medium may be coupled to the processor such that the processor may read information from, and write information to, the storage medium. In the alternative, the storage medium may be integral to the processor. The processor and the storage medium may reside in an application specific integrated circuit (“ASIC”). In the alternative, the processor and the 20 storage medium may reside as discrete components.

[0097] FIG. 7A illustrates a process 700 of a new block being added to a distributed ledger 720, according to example embodiments, and FIG. 7B illustrates contents of a new data block structure 730 for blockchain, according to example embodiments. Referring to FIG. 7A, clients (not shown) may submit transactions to blockchain nodes 711, 712, and/or 713. Clients may be instructions 25 received from any source to enact activity on the blockchain 720. As an example, clients may be

applications that act on behalf of a requester, such as a device, person or entity to propose transactions for the blockchain. The plurality of blockchain peers (*e.g.*, blockchain nodes 711, 712, and 713) may maintain a state of the blockchain network and a copy of the distributed ledger 720. Different types of blockchain nodes/peers may be present in the blockchain network including

5 endorsing peers which simulate and endorse transactions proposed by clients and committing peers which verify endorsements, validate transactions, and commit transactions to the distributed ledger 720. In this example, the blockchain nodes 711, 712, and 713 may perform the role of endorser node, committer node, or both.

[0098] The distributed ledger 720 includes a blockchain which stores immutable, sequenced

10 records in blocks, and a state database 724 (current world state) maintaining a current state of the blockchain 722. One distributed ledger 720 may exist per channel and each peer maintains its own copy of the distributed ledger 720 for each channel of which they are a member. The blockchain 722 is a transaction log, structured as hash-linked blocks where each block contains a sequence of N transactions. Blocks may include various components such as shown in FIG. 7B. The linking of the

15 blocks (shown by arrows in FIG. 7A) may be generated by adding a hash of a prior block's header within a block header of a current block. In this way, all transactions on the blockchain 722 are sequenced and cryptographically linked together preventing tampering with blockchain data without breaking the hash links. Furthermore, because of the links, the latest block in the blockchain 722 represents every transaction that has come before it. The blockchain 722 may be stored on a peer file

20 system (local or attached storage), which supports an append-only blockchain workload.

[0099] The current state of the blockchain 722 and the distributed ledger 722 may be stored in the state database 724. Here, the current state data represents the latest values for all keys ever included in the chain transaction log of the blockchain 722. Chaincode invocations execute transactions against the current state in the state database 724. To make these chaincode interactions extremely

25 efficient, the latest values of all keys are stored in the state database 724. The state database 724

may include an indexed view into the transaction log of the blockchain 722, it can therefore be regenerated from the chain at any time. The state database 724 may automatically get recovered (or generated if needed) upon peer startup, before transactions are accepted.

[00100] Endorsing nodes receive transactions from clients and endorse the transaction based on simulated results. Endorsing nodes hold smart contracts which simulate the transaction proposals. When an endorsing node endorses a transaction, the endorsing nodes creates a transaction endorsement which is a signed response from the endorsing node to the client application indicating the endorsement of the simulated transaction. The method of endorsing a transaction depends on an endorsement policy which may be specified within chaincode. An example of an endorsement policy is “the majority of endorsing peers must endorse the transaction”. Different channels may have different endorsement policies. Endorsed transactions are forward by the client application to ordering service 710.

[00101] The ordering service 710 accepts endorsed transactions, orders them into a block, and delivers the blocks to the committing peers. For example, the ordering service 710 may initiate a new block when a threshold of transactions has been reached, a timer times out, or another condition. In the example of FIG. 7A, blockchain node 712 is a committing peer that has received a new data new data block 730 for storage on blockchain 720. The first block in the blockchain may be referred to as a genesis block which includes information about the blockchain, its members, the data stored therein, etc.

[00102] The ordering service 710 may be made up of a cluster of orderers. The ordering service 710 does not process transactions, smart contracts, or maintain the shared ledger. Rather, the ordering service 710 may accept the endorsed transactions and specifies the order in which those transactions are committed to the distributed ledger 720. The architecture of the blockchain network may be designed such that the specific implementation of ‘ordering’ (e.g., Solo, Kafka, BFT, etc.) becomes a pluggable component.

[00103] Transactions are written to the distributed ledger 720 in a consistent order. The order of transactions is established to ensure that the updates to the state database 724 are valid when they are committed to the network. Unlike a cryptocurrency blockchain system (*e.g.*, Bitcoin, etc.) where ordering occurs through the solving of a cryptographic puzzle, or mining, in this example the parties
5 of the distributed ledger 720 may choose the ordering mechanism that best suits that network.

[00104] When the ordering service 710 initializes a new data block 730, the new data block 730 may be broadcast to committing peers (*e.g.*, blockchain nodes 711, 712, and 713). In response, each committing peer validates the transaction within the new data block 730 by checking to make sure that the read set and the write set still match the current world state in the state database 724.
10 Specifically, the committing peer can determine whether the read data that existed when the endorsers simulated the transaction is identical to the current world state in the state database 724. When the committing peer validates the transaction, the transaction is written to the blockchain 722 on the distributed ledger 720, and the state database 724 is updated with the write data from the read-write set. If a transaction fails, that is, if the committing peer finds that the read-write set does
15 not match the current world state in the state database 724, the transaction ordered into a block will still be included in that block, but it will be marked as invalid, and the state database 724 will not be updated.

[00105] Referring to FIG. 7B, a new data block 730 (also referred to as a data block) that is stored on the blockchain 722 of the distributed ledger 720 may include multiple data segments such as a
20 block header 740, block data 750 (block data section), and block metadata 760. It should be appreciated that the various depicted blocks and their contents, such as new data block 730 and its contents, shown in FIG. 7B are merely examples and are not meant to limit the scope of the example embodiments. In a conventional block, the data section may store transactional information of N transaction(s) (*e.g.*, 1, 10, 100, 500, 1000, 2000, 3000, etc.) within the block data 750.

[00106] The new data block 730 may include a link to a previous block (*e.g.*, on the blockchain 722 in FIG. 7A) within the block header 740. In particular, the block header 740 may include a hash of a previous block's header. The block header 740 may also include a unique block number, a hash of the block data 750 of the new data block 730, and the like. The block number of the new data block
5 730 may be unique and assigned in various orders, such as an incremental/sequential order starting from zero.

[00107] The block metadata 760 may store multiple fields of metadata (*e.g.*, as a byte array, etc.). Metadata fields may include signature on block creation, a reference to a last configuration block, a transaction filter identifying valid and invalid transactions within the block, last offset persisted of
10 an ordering service that ordered the block, and the like. The signature, the last configuration block, and the orderer metadata may be added by the ordering service 710. Meanwhile, a committer of the block (such as blockchain node 712) may add validity/invalidity information based on an endorsement policy, verification of read/write sets, and the like. The transaction filter may include a byte array of a size equal to the number of transactions that are included in the block data 750 and a
15 validation code identifying whether a transaction was valid/invalid.

[00108] FIG. 7C illustrates an embodiment of a blockchain 770 for digital content in accordance with the embodiments described herein. The digital content may include one or more files and associated information. The files may include media, images, video, audio, text, links, graphics, animations, web pages, documents, or other forms of digital content. The immutable, append-only
20 aspects of the blockchain serve as a safeguard to protect the integrity, validity, and authenticity of the digital content, making it suitable use in legal proceedings where admissibility rules apply or other settings where evidence is taken into consideration or where the presentation and use of digital information is otherwise of interest. In this case, the digital content may be referred to as digital evidence.

[00109] The blockchain may be formed in various ways. In one embodiment, the digital content may be included in and accessed from the blockchain itself. For example, each block of the blockchain may store a hash value of reference information (e.g., header, value, etc.) along the associated digital content. The hash value and associated digital content may then be encrypted together. Thus, the digital content of each block may be accessed by decrypting each block in the blockchain, and the hash value of each block may be used as a basis to reference a previous block. This may be illustrated as follows:

	<u>Block 1</u>	<u>Block 2</u>		<u>Block N</u>
	Hash Value 1	Hash Value 2		Hash Value N
10	Digital Content 1	Digital Content 2		Digital Content N

[00110] In one embodiment, the digital content may be not included in the blockchain. For example, the blockchain may store the encrypted hashes of the content of each block without any of the digital content. The digital content may be stored in another storage area or memory address in association with the hash value of the original file. The other storage area may be the same storage device used to store the blockchain or may be a different storage area or even a separate relational database. The digital content of each block may be referenced or accessed by obtaining or querying the hash value of a block of interest and then looking up that has value in the storage area, which is stored in correspondence with the actual digital content. This operation may be performed, for example, a database gatekeeper. This may be illustrated as follows:

20

25

<u>Blockchain</u>	<u>Storage Area</u>
Block 1 Hash Value	Block 1 Hash Value ... Content
.	.
.	.
5 .	.
Block N Hash Value	Block N Hash Value ... Content

[00111] In the example embodiment of FIG. 7C, the blockchain 770 includes a number of blocks 778₁, 778₂, ... 778_N cryptographically linked in an ordered sequence, where $N \geq 1$. The encryption
 10 used to link the blocks 778₁, 778₂, ... 778_N may be any of a number of keyed or un-keyed Hash functions. In one embodiment, the blocks 778₁, 778₂, ... 778_N are subject to a hash function which produces n-bit alphanumeric outputs (where n is 256 or another number) from inputs that are based on information in the blocks. Examples of such a hash function include, but are not limited to, a
 15 SHA-type (SHA stands for Secured Hash Algorithm) algorithm, Merkle-Damgard algorithm, HAIFA algorithm, Merkle-tree algorithm, nonce-based algorithm, and a non-collision-resistant PRF algorithm. In another embodiment, the blocks 778₁, 778₂, ... 778_N may be cryptographically linked by a function that is different from a hash function. For purposes of illustration, the following description is made with reference to a hash function, e.g., SHA-2.

[00112] Each of the blocks 778₁, 778₂, ..., 778_N in the blockchain includes a header, a version of
 20 the file, and a value. The header and the value are different for each block as a result of hashing in the blockchain. In one embodiment, the value may be included in the header. As described in greater detail below, the version of the file may be the original file or a different version of the original file.

[00113] The first block 778₁ in the blockchain is referred to as the genesis block and includes the
 25 header 772₁, original file 774₁, and an initial value 776₁. The hashing scheme used for the genesis block, and indeed in all subsequent blocks, may vary. For example, all the information in the first

block 778₁ may be hashed together and at one time, or each or a portion of the information in the first block 778₁ may be separately hashed and then a hash of the separately hashed portions may be performed.

[00114] The header 772₁ may include one or more initial parameters, which, for example, may include a version number, timestamp, nonce, root information, difficulty level, consensus protocol, duration, media format, source, descriptive keywords, and/or other information associated with original file 774₁ and/or the blockchain. The header 772₁ may be generated automatically (e.g., by blockchain network managing software) or manually by a blockchain participant. Unlike the header in other blocks 778₂ to 778_N in the blockchain, the header 772₁ in the genesis block does not
10 reference a previous block, simply because there is no previous block.

[00115] The original file 774₁ in the genesis block may be, for example, data as captured by a device with or without processing prior to its inclusion in the blockchain. The original file 774₁ is received through the interface of the system from the device, media source, or node. The original file 774₁ is associated with metadata, which, for example, may be generated by a user, the device,
15 and/or the system processor, either manually or automatically. The metadata may be included in the first block 778₁ in association with the original file 774₁.

[00116] The value 776₁ in the genesis block is an initial value generated based on one or more unique attributes of the original file 774₁. In one embodiment, the one or more unique attributes may include the hash value for the original file 774₁, metadata for the original file 774₁, and other
20 information associated with the file. In one implementation, the initial value 776₁ may be based on the following unique attributes:

- 1) SHA-2 computed hash value for the original file
- 2) originating device ID
- 3) starting timestamp for the original file
- 25 4) initial storage location of the original file

- 5) blockchain network member ID for software to currently control the original file and associated metadata

[00117] The other blocks 778_2 to 778_N in the blockchain also have headers, files, and values.

However, unlike the first block 772_1 , each of the headers 772_2 to 772_N in the other blocks includes the hash value of an immediately preceding block. The hash value of the immediately preceding block may be just the hash of the header of the previous block or may be the hash value of the entire previous block. By including the hash value of a preceding block in each of the remaining blocks, a trace can be performed from the Nth block back to the genesis block (and the associated original file) on a block-by-block basis, as indicated by arrows 780, to establish an auditable and immutable chain-of-custody.

[00118] Each of the header 772_2 to 772_N in the other blocks may also include other information, e.g., version number, timestamp, nonce, root information, difficulty level, consensus protocol, and/or other parameters or information associated with the corresponding files and/or the blockchain in general.

[00119] The files 774_2 to 774_N in the other blocks may be equal to the original file or may be a modified version of the original file in the genesis block depending, for example, on the type of processing performed. The type of processing performed may vary from block to block. The processing may involve, for example, any modification of a file in a preceding block, such as redacting information or otherwise changing the content of, taking information away from, or adding or appending information to the files.

[00120] Additionally, or alternatively, the processing may involve merely copying the file from a preceding block, changing a storage location of the file, analyzing the file from one or more preceding blocks, moving the file from one storage or memory location to another, or performing action relative to the file of the blockchain and/or its associated metadata. Processing which involves

analyzing a file may include, for example, appending, including, or otherwise associating various analytics, statistics, or other information associated with the file.

[00121] The values in each of the other blocks 776_2 to 776_N in the other blocks are unique values and are all different as a result of the processing performed. For example, the value in any one block corresponds to an updated version of the value in the previous block. The update is reflected in the hash of the block to which the value is assigned. The values of the blocks therefore provide an indication of what processing was performed in the blocks and also permit a tracing through the blockchain back to the original file. This tracking confirms the chain-of-custody of the file throughout the entire blockchain.

[00122] For example, consider the case where portions of the file in a previous block are redacted, blocked out, or pixelated in order to protect the identity of a person shown in the file. In this case, the block including the redacted file will include metadata associated with the redacted file, e.g., how the redaction was performed, who performed the redaction, timestamps where the redaction(s) occurred, etc. The metadata may be hashed to form the value. Because the metadata for the block is different from the information that was hashed to form the value in the previous block, the values are different from one another and may be recovered when decrypted.

[00123] In one embodiment, the value of a previous block may be updated (e.g., a new hash value computed) to form the value of a current block when any one or more of the following occurs. The new hash value may be computed by hashing all or a portion of the information noted below, in this example embodiment.

- a) new SHA-2 computed hash value if the file has been processed in any way (e.g., if the file was redacted, copied, altered, accessed, or some other action was taken)
- b) new storage location for the file
- c) new metadata identified associated with the file

- d) transfer of access or control of the file from one blockchain participant to another blockchain participant

[00124] FIG. 7D illustrates an embodiment of a block which may represent the structure of the blocks in the blockchain 790 in accordance with one embodiment. The block, Block_i, includes a header 772_i, a file 774_i, and a value 776_i.

[00125] The header 772_i includes a hash value of a previous block Block_{i-1} and additional reference information, which, for example, may be any of the types of information (e.g., header information including references, characteristics, parameters, etc.) discussed herein. All blocks reference the hash of a previous block except, of course, the genesis block. The hash value of the previous block may be just a hash of the header in the previous block or a hash of all or a portion of the information in the previous block, including the file and metadata.

[00126] The file 774_i includes a plurality of data, such as Data 1, Data 2, ..., Data N in sequence. The data are tagged with Metadata 1, Metadata 2, ..., Metadata N which describe the content and/or characteristics associated with the data. For example, the metadata for each data may include information to indicate a timestamp for the data, process the data, keywords indicating the persons or other content depicted in the data, and/or other features that may be helpful to establish the validity and content of the file as a whole, and particularly its use as digital evidence, for example, as described in connection with an embodiment discussed below. In addition to the metadata, each data may be tagged with reference REF₁, REF₂, ..., REF_N to a previous data to prevent tampering, gaps in the file, and sequential reference through the file.

[00127] Once the metadata is assigned to the data (e.g., through a smart contract), the metadata cannot be altered without the hash changing, which can easily be identified for invalidation. The metadata, thus, creates a data log of information that may be accessed for use by participants in the blockchain.

[00128] The value 776_i is a hash value or other value computed based on any of the types of information previously discussed. For example, for any given block Block_i , the value for that block may be updated to reflect the processing that was performed for that block, e.g., new hash value, new storage location, new metadata for the associated file, transfer of control or access, identifier, or other action or information to be added. Although the value in each block is shown to be separate from the metadata for the data of the file and header, the value may be based, in part or whole, on this metadata in another embodiment.

[00129] Once the blockchain 770 is formed, at any point in time, the immutable chain-of-custody for the file may be obtained by querying the blockchain for the transaction history of the values across the blocks. This query, or tracking procedure, may begin with decrypting the value of the block that is most currently included (e.g., the last (N^{th}) block), and then continuing to decrypt the value of the other blocks until the genesis block is reached and the original file is recovered. The decryption may involve decrypting the headers and files and associated metadata at each block, as well.

[00130] Decryption is performed based on the type of encryption that took place in each block. This may involve the use of private keys, public keys, or a public key-private key pair. For example, when asymmetric encryption is used, blockchain participants or a processor in the network may generate a public key and private key pair using a predetermined algorithm. The public key and private key are associated with each other through some mathematical relationship. The public key may be distributed publicly to serve as an address to receive messages from other users, e.g., an IP address or home address. The private key is kept secret and used to digitally sign messages sent to other blockchain participants. The signature is included in the message so that the recipient can verify using the public key of the sender. This way, the recipient can be sure that only the sender could have sent this message.

[00131] Generating a key pair may be analogous to creating an account on the blockchain, but without having to actually register anywhere. Also, every transaction that is executed on the blockchain is digitally signed by the sender using their private key. This signature ensures that only the owner of the account can track and process (if within the scope of permission determined by a smart contract) the file of the blockchain.

[00132] FIGS. 8A and 8B illustrate additional examples of use cases for blockchain which may be incorporated and used herein. In particular, FIG. 8A illustrates an example 800 of a blockchain 810 which stores machine learning (artificial intelligence) data. Machine learning relies on vast quantities of historical data (or training data) to build predictive models for accurate prediction on new data. Machine learning software (*e.g.*, neural networks, etc.) can often sift through millions of records to unearth non-intuitive patterns.

[00133] In the example of FIG. 8A, a host platform 820 builds and deploys a machine learning model for predictive monitoring of assets 830. Here, the host platform 820 may be a cloud platform, an industrial server, a web server, a personal computer, a user device, and the like. Assets 830 can be any type of asset (*e.g.*, machine or equipment, etc.) such as an aircraft, locomotive, turbine, medical machinery and equipment, oil and gas equipment, boats, ships, vehicles, and the like. As another example, assets 830 may be non-tangible assets such as stocks, currency, digital coins, insurance, or the like.

[00134] The blockchain 810 can be used to significantly improve both a training process 802 of the machine learning model and a predictive process 804 based on a trained machine learning model. For example, in 802, rather than requiring a data scientist / engineer or other user to collect the data, historical data may be stored by the assets 830 themselves (or through an intermediary, not shown) on the blockchain 810. This can significantly reduce the collection time needed by the host platform 820 when performing predictive model training. For example, using smart contracts, data can be directly and reliably transferred straight from its place of origin to the blockchain 810. By using the

blockchain 810 to ensure the security and ownership of the collected data, smart contracts may directly send the data from the assets to the individuals that use the data for building a machine learning model. This allows for sharing of data among the assets 830.

[00135] The collected data may be stored in the blockchain 810 based on a consensus mechanism.

5 The consensus mechanism pulls in (permissioned nodes) to ensure that the data being recorded is verified and accurate. The data recorded is time-stamped, cryptographically signed, and immutable. It is therefore auditable, transparent, and secure. Adding IoT devices which write directly to the blockchain can, in certain cases (i.e. supply chain, healthcare, logistics, etc.), increase both the frequency and accuracy of the data being recorded.

10 **[00136]** Furthermore, training of the machine learning model on the collected data may take rounds of refinement and testing by the host platform 820. Each round may be based on additional data or data that was not previously considered to help expand the knowledge of the machine learning model. In 802, the different training and testing steps (and the data associated therewith) may be stored on the blockchain 810 by the host platform 820. Each refinement of the machine learning
15 model (e.g., changes in variables, weights, etc.) may be stored on the blockchain 810. This provides verifiable proof of how the model was trained and what data was used to train the model.

Furthermore, when the host platform 820 has achieved a finally trained model, the resulting model may be stored on the blockchain 810.

[00137] After the model has been trained, it may be deployed to a live environment where it can
20 make predictions / decisions based on the execution of the final trained machine learning model. For example, in 804, the machine learning model may be used for condition-based maintenance (CBM) for an asset such as an aircraft, a wind turbine, a healthcare machine, and the like. In this example, data fed back from the asset 830 may be input the machine learning model and used to make event predictions such as failure events, error codes, and the like. Determinations made by the
25 execution of the machine learning model at the host platform 820 may be stored on the blockchain

810 to provide auditable / verifiable proof. As one non-limiting example, the machine learning model may predict a future breakdown/failure to a part of the asset 830 and create alert or a notification to replace the part. The data behind this decision may be stored by the host platform 820 on the blockchain 810. In one embodiment the features and/or the actions described and/or depicted
5 herein can occur on or with respect to the blockchain 810.

[00138] New transactions for a blockchain can be gathered together into a new block and added to an existing hash value. This is then encrypted to create a new hash for the new block. This is added to the next list of transactions when they are encrypted, and so on. The result is a chain of blocks that each contain the hash values of all preceding blocks. Computers that store these blocks
10 regularly compare their hash values to ensure that they are all in agreement. Any computer that does not agree, discards the records that are causing the problem. This approach is good for ensuring tamper-resistance of the blockchain, but it is not perfect.

[00139] One way to game this system is for a dishonest user to change the list of transactions in their favor, but in a way that leaves the hash unchanged. This can be done by brute force, in other
15 words by changing a record, encrypting the result, and seeing whether the hash value is the same. And if not, trying again and again and again until it finds a hash that matches. The security of blockchains is based on the belief that ordinary computers can only perform this kind of brute force attack over time scales that are entirely impractical, such as the age of the universe. By contrast, quantum computers are much faster (1000s of times faster) and consequently pose a much greater
20 threat.

[00140] FIG. 8B illustrates an example 850 of a quantum-secure blockchain 852 which implements quantum key distribution (QKD) to protect against a quantum computing attack. In this example, blockchain users can verify each other's identities using QKD. This sends information using quantum particles such as photons, which cannot be copied by an eavesdropper without destroying
25 them. In this way, a sender and a receiver through the blockchain can be sure of each other's identity.

[00141] In the example of FIG. 8B, four users are present 854, 856, 858, and 860. Each of pair of users may share a secret key 862 (*i.e.*, a QKD) between themselves. Since there are four nodes in this example, six pairs of nodes exists, and therefore six different secret keys 862 are used including QKD_{AB}, QKD_{AC}, QKD_{AD}, QKD_{BC}, QKD_{BD}, and QKD_{CD}. Each pair can create a QKD by sending
5 information using quantum particles such as photons, which cannot be copied by an eavesdropper without destroying them. In this way, a pair of users can be sure of each other's identity.

[00142] The operation of the blockchain 852 is based on two procedures (i) creation of transactions, and (ii) construction of blocks that aggregate the new transactions. New transactions may be created similar to a traditional blockchain network. Each transaction may contain information about a
10 sender, a receiver, a time of creation, an amount (or value) to be transferred, a list of reference transactions that justifies the sender has funds for the operation, and the like. This transaction record is then sent to all other nodes where it is entered into a pool of unconfirmed transactions. Here, two parties (*i.e.*, a pair of users from among 854-860) authenticate the transaction by providing their shared secret key 862 (QKD). This quantum signature can be attached to every
15 transaction making it exceedingly difficult to tamper with. Each node checks their entries with respect to a local copy of the blockchain 852 to verify that each transaction has sufficient funds. However, the transactions are not yet confirmed.

[00143] Rather than perform a traditional mining process on the blocks, the blocks may be created in a decentralized manner using a broadcast protocol. At a predetermined period of time (*e.g.*,
20 seconds, minutes, hours, etc.) the network may apply the broadcast protocol to any unconfirmed transaction thereby to achieve a Byzantine agreement (consensus) regarding a correct version of the transaction. For example, each node may possess a private value (transaction data of that particular node). In a first round, nodes transmit their private values to each other. In subsequent rounds, nodes communicate the information they received in the previous round from other nodes. Here,
25 honest nodes are able to create a complete set of transactions within a new block. This new block

can be added to the blockchain 852. In one embodiment the features and/or the actions described and/or depicted herein can occur on or with respect to the blockchain 852.

[00144] FIG. 9 illustrates an example system 900 that supports one or more of the example embodiments described and/or depicted herein. The system 900 comprises a computer system/server 902, which is operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well-known computing systems, environments, and/or configurations that may be suitable for use with computer system/server 902 include, but are not limited to, personal computer systems, server computer systems, thin clients, thick clients, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputer systems, mainframe computer systems, and distributed cloud computing environments that include any of the above systems or devices, and the like.

[00145] Computer system/server 902 may be described in the general context of computer system-executable instructions, such as program modules, being executed by a computer system. Generally, program modules may include routines, programs, objects, components, logic, data structures, and so on that perform particular tasks or implement particular abstract data types. Computer system/server 902 may be practiced in distributed cloud computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed cloud computing environment, program modules may be located in both local and remote computer system storage media including memory storage devices.

[00146] As shown in FIG. 9, computer system/server 902 in cloud computing node 900 is shown in the form of a general-purpose computing device. The components of computer system/server 902 may include, but are not limited to, one or more processors or processing units 904, a system memory 906, and a bus that couples various system components including system memory 906 to processor 904.

[00147] The bus represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnects (PCI) bus.

[00148] Computer system/server 902 typically includes a variety of computer system readable media. Such media may be any available media that is accessible by computer system/server 902, and it includes both volatile and non-volatile media, removable and non-removable media. System memory 906, in one embodiment, implements the flow diagrams of the other figures. The system memory 906 can include computer system readable media in the form of volatile memory, such as random-access memory (RAM) 910 and/or cache memory 912. Computer system/server 902 may further include other removable/non-removable, volatile/non-volatile computer system storage media. By way of example only, storage system 914 can be provided for reading from and writing to a non-removable, non-volatile magnetic media (not shown and typically called 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”), and an optical disk drive for reading from or writing to a removable, non-volatile optical disk such as a CD-ROM, DVD-ROM or other optical media can be provided. In such instances, each can be connected to the bus by one or more data media interfaces. As will be further depicted and described below, memory 906 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 of the application.

[00149] Program/utility 916, having a set (at least one) of program modules 918, may be stored in memory 906 by way of example, and not limitation, as well as an operating system, one or more application programs, other program modules, and program data. Each of the operating system, one

or more application programs, other program modules, and program data or some combination thereof, may include an implementation of a networking environment. Program modules 918 generally carry out the functions and/or methodologies of various embodiments of the application as described herein.

5 **[00150]** As will be appreciated by one skilled in the art, aspects of the present application may be embodied as a system, method, or computer program product. Accordingly, aspects of the present application may take the form of an entirely hardware embodiment, an entirely software embodiment (including firmware, resident software, micro-code, etc.) or an embodiment combining software and hardware aspects that may all generally be referred to herein as a “circuit,” “module”
10 or “system.” Furthermore, aspects of the present application may take the form of a computer program product embodied in one or more computer readable medium(s) having computer readable program code embodied thereon.

[00151] Computer system/server 902 may also communicate with one or more external devices 920 such as a keyboard, a pointing device, a display 922, etc.; one or more devices that enable a user to
15 interact with computer system/server 902; and/or any devices (e.g., network card, modem, etc.) that enable computer system/server 902 to communicate with one or more other computing devices. Such communication can occur via I/O interfaces 924. Still yet, computer system/server 902 can communicate with one or more networks such as a local area network (LAN), a general wide area network (WAN), and/or a public network (e.g., the Internet) via network adapter 926. As depicted,
20 network adapter 926 communicates with the other components of computer system/server 902 via a bus. It should be understood that although not shown, other hardware and/or software components could be used in conjunction with computer system/server 902. Examples include, but are not limited to: microcode, device drivers, redundant processing units, external disk drive arrays, RAID systems, tape drives, and data archival storage systems, etc.

[00152] Although an exemplary embodiment of at least one of a system, method, and non-transitory computer readable medium has been illustrated in the accompanied drawings and described in the foregoing detailed description, it will be understood that the application is not limited to the embodiments disclosed, but is capable of numerous rearrangements, modifications, and substitutions as set forth and defined by the following claims. For example, the capabilities of the system of the various figures can be performed by one or more of the modules or components described herein or in a distributed architecture and may include a transmitter, receiver or pair of both. For example, all or part of the functionality performed by the individual modules, may be performed by one or more of these modules. Further, the functionality described herein may be performed at various times and in relation to various events, internal or external to the modules or components. Also, the information sent between various modules can be sent between the modules via at least one of: a data network, the Internet, a voice network, an Internet Protocol network, a wireless device, a wired device and/or via plurality of protocols. Also, the messages sent or received by any of the modules may be sent or received directly and/or via one or more of the other modules.

[00153] One skilled in the art will appreciate that a “system” could be embodied as a personal computer, a server, a console, a personal digital assistant (PDA), a cell phone, a tablet computing device, a smartphone or any other suitable computing device, or combination of devices. Presenting the above-described functions as being performed by a “system” is not intended to limit the scope of the present application in any way but is intended to provide one example of many embodiments.

Indeed, methods, systems and apparatuses disclosed herein may be implemented in localized and distributed forms consistent with computing technology.

[00154] It should be noted that some of the system features described in this specification have been presented as modules, in order to more particularly emphasize their implementation independence. For example, a module may be implemented as a hardware circuit comprising custom very large-scale integration (VLSI) circuits or gate arrays, off-the-shelf semiconductors such as

logic chips, transistors, or other discrete components. A module may also be implemented in programmable hardware devices such as field programmable gate arrays, programmable array logic, programmable logic devices, graphics processing units, or the like.

[00155] A module may also be at least partially implemented in software for execution by various
5 types of processors. An identified unit of executable code may, for instance, comprise one or more physical or logical blocks of computer instructions that may, for instance, be organized as an object, procedure, or function. Nevertheless, the executables of an identified module need not be physically located together but may comprise disparate instructions stored in different locations which, when joined logically together, comprise the module and achieve the stated purpose for the module.

10 Further, modules may be stored on a computer-readable medium, which may be, for instance, a hard disk drive, flash device, random access memory (RAM), tape, or any other such medium used to store data.

[00156] Indeed, a module of executable code could be a single instruction, or many instructions, and may even be distributed over several different code segments, among different programs, and
15 across several memory devices. Similarly, operational data may be identified and illustrated herein within modules and may be embodied in any suitable form and organized within any suitable type of data structure. The operational data may be collected as a single data set or may be distributed over different locations including over different storage devices, and may exist, at least partially, merely as electronic signals on a system or network.

20 [00157] It will be readily understood that the components of the application, as generally described and illustrated in the figures herein, may be arranged and designed in a wide variety of different configurations. Thus, the detailed description of the embodiments is not intended to limit the scope of the application as claimed but is merely representative of selected embodiments of the application.

[00158] One having ordinary skill in the art will readily understand that the above may be practiced
25 with steps in a different order, and/or with hardware elements in configurations that are different

than those which are disclosed. Therefore, although the application has been described based upon these preferred embodiments, it would be apparent to those of skill in the art that certain modifications, variations, and alternative constructions would be apparent.

[00159] While preferred embodiments of the present application have been described, it is to be understood that the embodiments described are illustrative only and the scope of the application is to be defined solely by the appended claims when considered with a full range of equivalents and modifications (e.g., protocols, hardware devices, software platforms etc.) thereto.

What is claimed is:

1. An apparatus, comprising:

a hardware processor configured to:

generate a block comprising a plurality of blockchain transactions;

identify a blockchain transaction, among the plurality of blockchain transactions, in the block which comprises noisy key-value pairs stored therein and which are configured to change valid key-value pairs stored on a blockchain ledger to the noisy key-value pairs based on a tag value included in the block, wherein

the tag value identifies the blockchain transaction as a noisy blockchain transaction,

the block further includes a key identifier indicative of a secret key that encrypts the tag value, and

the identification as the noisy blockchain transaction includes decrypting the tag value based on the secret key indicated by the key identifier; and

execute an update to the blockchain ledger based on the plurality of blockchain transactions included in the block, wherein the hardware processor is configured to skip execution of the noisy blockchain transaction which changes the valid key-value pairs stored on the blockchain ledger to the noisy key-value pairs based on the identification.

2. The apparatus of claim 1, wherein the hardware processor is further configured to:

receive signatures and read-write sets with respect to the noisy blockchain transaction from endorser peers of the blockchain ledger; and

store the signatures and the read-write sets in the noisy blockchain transaction.

3. The apparatus of claim 1, wherein the hardware processor is further configured to generate the noisy blockchain transaction based on a historical pattern of behavior of multiple clients on the blockchain ledger.

4. The apparatus of claim 1, wherein the hardware processor is further configured to obscure the tag value based on the secret key prior to transmission of the noisy blockchain transaction to another peer node.

5. The apparatus of claim 4, wherein

the secret key comprises a public key of an asymmetric public and private key pair, and

a private key of the asymmetric public and private keypair is shared by blockchain clients within a blockchain network that created the noisy blockchain transaction but not shared by other blockchain clients within the blockchain network.

6. An apparatus, comprising:

a network interface configured to receive a block comprising a plurality of blockchain transactions; and

a hardware processor configured to:

identify a blockchain transaction, among the plurality of blockchain transactions, in the block that comprises noisy key-value pairs stored therein and which are configured to change valid key-value pairs stored on a blockchain ledger to the noisy key-value pairs based on a tag value, wherein

the tag value identifies the blockchain transaction as a noisy blockchain transaction,

the block further includes a key identifier indicative of a secret key that encrypts the tag value, and

the identification as the noisy blockchain transaction includes decrypting the tag value based on the secret key indicated by the key identifier; and

execute an update to the blockchain ledger based on the plurality of blockchain transactions included in the block, wherein the hardware processor is configured to skip execution of the noisy blockchain transaction which changes the valid key-value pairs stored on the blockchain ledger to the noisy key-value pairs based on the identification.

7. The apparatus of claim 6, wherein the hardware processor is further configured to:

verify signatures and read-write sets with respect to the noisy blockchain transaction from endorser peers of the noisy blockchain transaction; and

store the signatures and the read-write sets within a data block.

8. The apparatus of claim 6, wherein the tag value comprises an obscured tag value that has been obscured based on the secret key.

9. The apparatus of claim 8, wherein

the secret key comprises a public key of an asymmetric public and private key pair, and

a private key of the asymmetric public and private key pair is shared by blockchain clients within a blockchain network that created the noisy blockchain transaction but not shared by other blockchain clients within the blockchain network.

10. The apparatus of claim 6, wherein

the network interface is further configured to receive a non-noisy blockchain transaction that comprises live data for the blockchain ledger, and a specific tag value, wherein the specific tag value identifies that the non-noisy blockchain transaction is not noise.

11. A method, comprising:

generating, by a first blockchain peer, a blockchain transaction that comprises noisy key-value pairs stored therein which are configured to change valid key-value pairs stored on a blockchain ledger to the noisy key-value pairs;

generating, by the first blockchain peer, a tag value which identifies the blockchain transaction as a noisy blockchain transaction;

storing, by the first blockchain peer, the tag value within the blockchain transaction; and

transmitting, from the first blockchain peer, the blockchain transaction to one or more blockchain peers of the blockchain ledger different from the first blockchain peer;

receiving, by at least a second blockchain peer of the one or more blockchain peers, a block comprising a plurality of blockchain transactions including the blockchain transaction that is tagged as the noisy blockchain transaction;

identifying, by the at least second blockchain peer, the blockchain transaction, among the plurality of blockchain transactions, as the noisy blockchain transaction based on the tag value, wherein

the block further includes a key identifier indicative of a secret key that encrypts the tag value, and

the identifying of the blockchain transaction as the noisy blockchain transaction includes decrypting the tag value based on the secret key indicated by the key identifier; and

executing, by the at least second blockchain peer, an update to the blockchain ledger based on the plurality of blockchain transactions included in the block, wherein the executing comprises skipping execution of the noisy blockchain transaction which changes the valid key-value pairs stored on the blockchain ledger to the noisy key-value pairs based on the identification.

12. The method of claim 11, further comprising:

receiving signatures and read-write sets with respect to the noisy blockchain transaction from endorser peers of the blockchain ledger; and

storing the signatures and the read-write sets in the noisy blockchain transaction.

13. The method of claim 11, wherein the generating of the blockchain transaction comprises generating the blockchain transaction based on a historical pattern of behavior of multiple clients of the blockchain ledger.

14. The method of claim 11, wherein the generating of the tag value further comprises obscuring the tag value with the secret key prior to the transmitting of the blockchain transaction.