

## **MANAGING BLOCKCHAIN-BASED CENTRALIZED LEDGER SYSTEMS**

### **TECHNICAL FIELD**

**[0001]** This specification relates to managing blockchain-based centralized ledger systems.

### **BACKGROUND**

**[0002]** A ledger is typically used to record a history of transactions, such as economic and financial activities in an organization. Applications with ledger-like functionality, such as custom audit tables or audit trails created in relational databases, have been built to maintain an accurate history of applications' data. However, building such applications is time-consuming and prone to human error. Also, as the relational databases are not inherently immutable, any changes to the data are hard to track and verify.

**[0003]** Distributed ledger systems (DLSs), which can also be referred to as consensus networks, and/or blockchain networks, enable participating entities to securely, and immutably store data. DLSs are commonly referred to as blockchain networks without referencing any particular user case. Examples of types of blockchain networks can include public blockchain networks, private blockchain networks, and consortium blockchain networks. Blockchain networks perform a consensus process to validate each transaction before the transaction can be added to the blockchain networks, which can be time-consuming, ineffective, and complicated.

**[0004]** Therefore, it would be desirable to develop a ledger system that can effectively and securely manage transactions while providing immutability, reliability, trustworthiness, and verifiability of the transactions.

### **SUMMARY**

**[0005]** This specification describes technologies for managing blockchain-based centralized ledger systems. These technologies generally involve a blockchain-based centralized ledger system (e.g., a universal auditable ledger service system) that adopts a data structure of a blockchain to leverage immutability, reliability, and trustworthiness of data stored on the blockchain. The centralized ledger system can obtain trust timestamp information from a trust time server that is independent from the centralized ledger system

(e.g., a third-party, globally acknowledged time authority). The centralized ledger system can leverage the established trust on the timestamp information provided by the trust timer server and integrate the trust timestamp information into the centralized ledger system for the data stored on the blockchain, which can further enhance credibility, auditability, and legality of the stored data.

[0006] These technologies described herein can help reduce costs of obtaining trust timestamp information from the trust time server. For example, the described technologies can provide a cost-effective trust timestamp service for a large number of blocks in multiple blockchains maintained by the centralized ledger system.

[0007] This specification also provides one or more non-transitory computer-readable storage media coupled to one or more processors and having instructions stored thereon which, when executed by the one or more processors, cause the one or more processors to perform operations in accordance with embodiments of the methods provided herein.

[0008] This specification further provides a system for implementing the methods provided herein. The system includes one or more processors, and a computer-readable storage medium coupled to the one or more processors having instructions stored thereon which, when executed by the one or more processors, cause the one or more processors to perform operations in accordance with embodiments of the methods provided herein.

[0009] It is appreciated that methods in accordance with this specification may include any combination of the aspects and features described herein. That is, methods in accordance with this specification are not limited to the combinations of aspects and features specifically described herein, but also include any combination of the aspects and features provided.

[0010] The details of one or more embodiments of this specification are set forth in the accompanying drawings and the description below. Other features and advantages of this specification will be apparent from the description and drawings, and from the claims.

## **BRIEF DESCRIPTION OF THE DRAWINGS**

[0011] FIG. 1 is a diagram illustrating an example of an environment that can be used to execute embodiments of this specification.

[0012] FIG. 2 is a diagram illustrating an example of an architecture in accordance with embodiments of this specification.

[0013] FIG. 3 is a diagram illustrating an example of an environment implementing trust timestamp services in a blockchain-based centralized ledger system in accordance with embodiments of this specification.

[0014] FIG. 4A is a diagram illustrating an example of a blockchain-based centralized ledger system for implementing a trust timestamp service in a single ledger server associated with a single client in accordance with embodiments of this specification.

[0015] FIG. 4B is a diagram illustrating an example of a blockchain-based centralized ledger system for providing a trust timestamp service to multiple clients by a joint ledger server in accordance with embodiments of this specification.

[0016] FIG. 5 is a diagram illustrating an example of a process of maintaining a centralized trust timestamp blockchain in a blockchain-based centralized ledger system in accordance with embodiments of this specification.

[0017] FIG. 6 is a flowchart illustrating an example of a process that can be executed in accordance with embodiments of this specification.

[0018] FIG. 7 depicts examples of modules of an apparatus in accordance with embodiments of this specification.

[0019] Like reference numbers and designations in the various drawings indicate like elements.

## **DETAILED DESCRIPTION**

[0020] This specification describes technologies for managing blockchain-based centralized ledger systems. These technologies generally involve a blockchain-based centralized ledger system (e.g., a universal auditable ledger service system) that adopts a data structure of a blockchain to leverage immutability, reliability, and trustworthiness of data stored on the blockchain. The centralized ledger system can obtain trust timestamp information from a trust time server that is independent from the centralized ledger system (e.g., a third-party, globally acknowledged time authority). The centralized ledger system can leverage the established trust on the timestamp information provided by the trust timer server and integrate the trust timestamp information into the centralized ledger system for the data stored on the blockchain, which can further enhance credibility, auditability, and legality of the stored data.

**[0021]** These technologies described herein can help reduce costs of obtaining trust timestamp information from the trust time server. For example, the described technologies can provide a cost-effective trust timestamp service for a large number of blocks in multiple blockchains maintained by the centralized ledger system.

**[0022]** The techniques described in this specification produce several technical effects. In some embodiments, the blockchain-based centralized ledger system can be a ledger system based on centralization, which can provide a cryptographically verifiable state-independent data ledger storage with time-critical auditing (with non-repudiation and temper-resistance). In some embodiments, the blockchain-based centralized ledger system can provide ledger services based on a cloud platform featuring centralized endorsement with credibility and neutrality. The blockchain-based centralized ledger system can provide highly reliable and high-performance auditable streaming ledger services by leveraging both a blockchain system's high credibility and a centralized system's high performance and low latency for handling various types of data and logs with auditing requirements, traceability, and tracking.

**[0023]** In some embodiments, the blockchain-based centralized ledger system can include a central trusted authority that provides transparent, immutable, and cryptographically verifiable data that are stored in blocks of a blockchain data structure. In some embodiments, the stored data can be in a log format, including, for example, not only for transaction logs but also other transaction data and block data. Due to the existence of the central trusted authority, the blockchain-based centralized ledger system does not need to perform consensus processes to establish trust, which can result in significant time- and cost-saving. In some embodiments, the blockchain-based centralized ledger system can be more efficient compared to a typical blockchain-based distributed or decentralized ledger system. In some embodiments, the blockchain-based centralized ledger system can provide a cloud-based storage service with enhanced trust, efficiency, and storage performance.

**[0024]** In some embodiments, the blockchain-based centralized ledger system can enhance credibility, auditability, and legality of stored data on a blockchain. For example, the centralized ledger system can interface with a trust time server and provide trust timestamp information of the trust time server to clients of the blockchain-based centralized ledger system. The trust time server is independent from the centralized ledger system. The trust

time server can be associated with a third-party trust time authority that provides accurate time services and can be acknowledged or trusted, for example, globally, by the public, auditing entities (such as companies, institutions, or organizations), and/or legal entities (such as courts or governments). As the trustworthiness of the timestamp information provided by the trust time server is acknowledged, integrating the timestamp information of the trust time server into the centralized ledger system for the data stored on the blockchain can further enhance credibility, auditability, and legality of the data stored in the centralized ledger system.

**[0025]** In some embodiments, the blockchain-based centralized ledger system features respective rights of parties or participants of the blockchain-based centralized ledger system. For example, a client of the blockchain-based centralized ledger system can have a right of providing a signature for storing transaction data on a blockchain in the centralized ledger system such that the client cannot repudiate the transaction data. In some embodiments, the centralized ledger system has a right of providing a signature for acknowledging the storage of the transaction data such that the centralized ledger system cannot deny storing the transaction data. In some embodiments, a trust time server has a right of providing a signature for trust timestamp information for the transaction data stored on the centralized ledger system such that the trust time server cannot deny the trust timestamp information. In some embodiments, the three respective rights of the three parties (the client, the ledger system, and the trust time server) are independent from each other, which can further enhance creditability and auditability of the transaction data stored in the centralized ledger system.

**[0026]** In some embodiments, the blockchain-based centralized ledger system can provide enhanced orderliness and authenticity of transaction data stored on the blockchain. For example, the blockchain-based centralized ledger system can transmit trust timestamp requests for transaction data stored on a blockchain to the trust time server, and the trust time server can provide trust timestamp information such as timestamps and associated signatures, for example, to authenticate or endorse the time of the transaction data stored on a blockchain. The centralized ledger system can store the trust timestamp information, e.g., as transactions, in the blockchain. The trust timestamp information can be used to verify orderliness and authenticity of the transaction data stored on the blockchain, which in turn can provide

enhanced creditability, auditability and legality of the transaction data stored on the blockchain.

**[0027]** In some embodiments, the blockchain-based centralized ledger system can help reduce costs of obtaining trust timestamp information from the trust time server. For example, the blockchain-based centralized ledger system can provide a cost-effective trust timestamp service for a large number of blocks in multiple blockchains maintained by the centralized ledger system. The trust time server charges the centralized ledger system for providing trust timestamp information to each timestamp request, and thus the large number of timestamp requests can incur high cost on the centralized ledger system and clients. In some embodiments, the centralized ledger system can maintain a centralized trust timestamp blockchain by transmitting (e.g., periodically) server timestamp requests to the trust time server and storing a series of trust timestamps and associated signatures of the trust time server in a series of timestamp blocks. The centralized ledger server can generate the series of timestamp blocks each storing a respective trust timestamp and associated signature of the trust time server. The timestamp blocks can be anchored or linked together in the centralized trust timestamp blockchain. Two blocks can be anchored to each other by one of the blocks including information that uniquely identifies the other one of the blocks (e.g., a hash of the other one of the blocks), such that any changes made to the other one of the blocks can be detected by examining the information included in the one of the blocks. The centralized trust timestamp blockchain can be used to provide timestamp services to the blockchains in the centralized ledger system, which can avoid transmitting the large number of timestamp requests directly to the trust time server. In such a way, the centralized ledger system can reduce the overall cost for obtaining trust timestamp information from the trust time server.

**[0028]** In some embodiments, the blockchain-based centralized ledger system can reduce the number of timestamp requests transmitted to the trust time server. For example, the centralized ledger system can include multiple ledger servers each storing transaction data in a respective blockchain. Each ledger server can transmit (e.g., periodically) timestamp requests for blocks in the respective blockchain. Instead of directly transmitting the timestamp requests from the multiple ledger servers to the trust time server, the centralized ledger system can include a centralized ledger server in communication with the multiple ledger servers. The centralized ledger server can transmit (e.g., periodically) server

timestamp requests to the trust time server and maintain the centralized trust timestamp blockchain. The centralized trust timestamp blockchain can be used to provide timestamp services to multiple blockchains in the centralized ledger system. For example, when the centralized ledger server receives a timestamp request for a block in a blockchain from a ledger server associated with the blockchain, the centralized ledger server can transmit a most recent timestamp and associated signature stored in the centralized trust timestamp blockchain to the ledger server. In such a way, the centralized ledger system can greatly reduce the number of timestamp requests transmitted to the trust time server, thereby reducing the cost for obtaining trust timestamps from the trust time server.

**[0029]** In some embodiments, the blockchain-based centralized ledger system can provide enhanced orderliness and authenticity of transaction data stored on the blockchains. For example, a ledger server can store a timestamp and associated signature of the centralized trust timestamp blockchain in a particular timestamped block in a blockchain associated with the ledger server. Thus, any block preceding the particular timestamped block in the blockchain can be determined to be generated before a time point represented by the timestamp. Moreover, the timestamp blocks in the centralized trust timestamp blockchain are anchored or linked together. Multiple timestamped blocks storing respective timestamps and associated signatures of the centralized trust timestamp blockchain in a blockchain can be also anchored with each other according to the timestamps. One or more non-timestamped blocks can exist between adjacent two of the timestamped blocks and store no timestamp and associated signature from the trust time server. Thus, the centralized trust timestamp blockchain can guarantee that non-timestamped blocks in the blockchain between adjacent two timestamped blocks associated with adjacent two particular timestamps of the centralized trust timestamp blockchain are generated at time points between the adjacent two particular timestamps.

**[0030]** To provide further context for embodiments of this specification, and as introduced above, distributed ledger systems (DLSs), which can also be referred to as consensus networks (e.g., made up of peer-to-peer nodes), and blockchain networks, enable participating entities to securely, and immutably conduct transactions, and store data. Although the term blockchain is generally associated with particular networks, and/or use

cases, blockchain is used herein to generally refer to a DLS without reference to any particular use case.

**[0031]** A blockchain is a data structure that stores transactions in a way that the transactions are immutable. Thus, transactions recorded on a blockchain are reliable and trustworthy. A blockchain includes one or more blocks. Each block in the chain is linked to a previous block immediately before it in the chain by including a hash of the previous block. Each block also includes a local timestamp (e.g., provided by a computing device that generates the block or a computing system that manages the blockchain), its own hash, and one or more transactions. For example, the block can include a block header and a block body. The block header can include the local timestamp, its own hash, and a hash of the previous block. The block body can include payload information such as the one or more transactions (or transaction data). The transactions, which have already been verified by the nodes of the blockchain network, are hashed and encoded into a Merkle tree. A Merkle tree is a data structure in which data at the leaf nodes of the tree is hashed, and all hashes in each branch of the tree are concatenated at the root of the branch. This process continues up the tree to the root of the entire tree, which stores a hash that is representative of all data in the tree. A hash purporting to be of a transaction stored in the tree can be quickly verified by determining whether it is consistent with the structure of the tree.

**[0032]** Whereas a blockchain is a decentralized or at least partially decentralized data structure for storing transactions, a blockchain network is a network of computing nodes that manage, update, and maintain one or more blockchains by broadcasting, verifying and validating transactions, etc. As introduced above, a blockchain network can be provided as a public blockchain network, a private blockchain network, or a consortium blockchain network.

**[0033]** In general, a consortium blockchain network is private among the participating entities. In a consortium blockchain network, the consensus process is controlled by an authorized set of nodes, which can be referred to as consensus nodes, one or more consensus nodes being operated by a respective entity (e.g., a financial institution, insurance company). For example, a consortium of ten (10) entities (e.g., financial institutions, insurance companies) can operate a consortium blockchain network, each of which operates at least one node in the consortium blockchain network. In some examples, within a consortium

blockchain network, a global blockchain is provided as a blockchain that is replicated across all nodes. That is, all consensus nodes are in perfect state consensus with respect to the global blockchain. To achieve consensus (e.g., agreement to the addition of a block to a blockchain), a consensus protocol is implemented within the consortium blockchain network. For example, the consortium blockchain network can implement a practical Byzantine fault tolerance (PBFT) consensus, described in further detail below.

**[0034]** In some embodiments, a centralized ledger system can also adopt the data structure of a blockchain to leverage immutability, reliability, and trustworthiness of data stored on a blockchain. In some embodiments, such a centralized ledger system can be referred to as a blockchain-based centralized ledger system or a universal auditable ledger service system. In some embodiments, the blockchain-based centralized ledger system can include a central trusted authority that provides transparent, immutable, and cryptographically verifiable data that are stored in blocks of a blockchain data structure. The stored data can be in a log format, including, for example, not only for transaction logs but also other transaction data and block data. Due to the existence of the central trusted authority, the blockchain-based centralized ledger system does not need to perform consensus processes to establish trust. In some embodiments, the blockchain-based centralized ledger system can be more efficient compared to a typical blockchain-based distributed or decentralized ledger system. In some embodiments, the blockchain-based centralized ledger system can provide a cloud-based storage service with enhanced trust, efficiency, and storage performance.

**[0035]** In some embodiments, the centralized ledger system can be a node of a blockchain network. For example, the centralized ledger system can be a non-consensus node in the blockchain network and can provide highly reliable and high-performance auditable streaming ledger services for the consensus nodes or other non-consensus nodes in the blockchain network, or entities outside of the blockchain network.

**[0036]** FIG. 1 is a diagram illustrating an example of an environment 100 that can be used to execute embodiments of this specification. In some examples, the environment 100 enables entities to participate in a consortium blockchain network 102. The environment 100 includes computing devices 106, 108, and a network 110. In some examples, the network 110 includes a local area network (LAN), wide area network (WAN), the Internet, or a

combination thereof, and connects web sites, user devices (e.g., computing devices), and back-end systems. In some examples, the network 110 can be accessed over a wired and/or a wireless communications link. In some examples, the network 110 enables communication with, and within the consortium blockchain network 102. In general the network 110 represents one or more communication networks. In some cases, the computing devices 106, 108 can be nodes of a cloud computing system (not shown), or each computing device 106, 108 can be a separate cloud computing system including a number of computers interconnected by a network and functioning as a distributed processing system.

**[0037]** In the depicted example, the computing devices 106, 108 can each include any appropriate computing system that enables participation as a node in the consortium blockchain network 102. Examples of computing devices include, without limitation, a server, a desktop computer, a laptop computer, a tablet computing device, and a smartphone. In some examples, the computing devices 106, 108 host one or more computer-implemented services for interacting with the consortium blockchain network 102. For example, the computing device 106 can host computer-implemented services of a first entity (e.g., user A), such as a transaction management system that the first entity uses to manage its transactions with one or more other entities (e.g., other users). The computing device 108 can host computer-implemented services of a second entity (e.g., user B), such as a transaction management system that the second entity uses to manage its transactions with one or more other entities (e.g., other users). In the example of FIG. 1, the consortium blockchain network 102 is represented as a peer-to-peer network of nodes, and the computing devices 106, 108 provide nodes of the first entity, and second entity respectively, which participate in the consortium blockchain network 102.

**[0038]** FIG. 2 is a diagram illustrating an example of an architecture 200 in accordance with embodiments of the specification. The example conceptual architecture 200 includes participant systems 202, 204, 206 that correspond to Participant A, Participant B, and Participant C, respectively. Each participant (e.g., user, enterprise) participates in a blockchain network 212 provided as a peer-to-peer network including multiple nodes 214, at least some of which immutably record information in a blockchain 216. Although a single blockchain 216 is schematically depicted within the blockchain network 212, multiple copies

of the blockchain 216 are provided, and are maintained across the blockchain network 212, as described in further detail herein.

**[0039]** In the depicted example, each participant system 202, 204, 206 is provided by, or on behalf of Participant A, Participant B, and Participant C, respectively, and functions as a respective node 214 within the blockchain network. As used herein, a node generally refers to an individual system (e.g., computer, server) that is connected to the blockchain network 212, and enables a respective participant to participate in the blockchain network. In the example of FIG. 2, a participant corresponds to each node 214. It is contemplated, however, that a participant can operate multiple nodes 214 within the blockchain network 212, and/or multiple participants can share a node 214. In some examples, the participant systems 202, 204, 206 communicate with, or through the blockchain network 212 using a protocol (e.g., hypertext transfer protocol secure (HTTPS)), and/or using remote procedure calls (RPCs).

**[0040]** Nodes 214 can have varying degrees of participation within the blockchain network 212. For example, some nodes 214 can participate in the consensus process (e.g., as miner nodes that add blocks to the blockchain 216), while other nodes 214 do not participate in the consensus process. As another example, some nodes 214 store a complete copy of the blockchain 216, while other nodes 214 only store copies of portions of the blockchain 216. For example, data access privileges can limit the blockchain data that a respective participant stores within its respective system. In the example of FIG. 2, the participant systems 202, 204, and 206 store respective, complete copies 216', 216'', and 216''' of the blockchain 216.

**[0041]** A blockchain (e.g., the blockchain 216 of FIG. 2) is made up of a chain of blocks, each block storing data. Examples of data include transaction data representative of a transaction between two or more participants. While transactions are used herein by way of non-limiting example, it is contemplated that any appropriate data can be stored in a blockchain (e.g., documents, images, videos, audio). Examples of a transaction can include, without limitation, exchanges of something of value (e.g., assets, products, services, currency). The transaction data is immutably stored within the blockchain. That is, the transaction data cannot be changed.

**[0042]** Before storing in a block, the transaction data is hashed. Hashing is a process of transforming the transaction data (provided as string data) into a fixed-length hash value (also provided as string data). It is not possible to un-hash the hash value to obtain the

transaction data. Hashing ensures that even a slight change in the transaction data results in a completely different hash value. Further, and as noted above, the hash value is of fixed length. That is, no matter the size of the transaction data the length of the hash value is fixed. Hashing includes processing the transaction data through a hash function to generate the hash value. An example of a hash function includes, without limitation, the secure hash algorithm (SHA)-256, which outputs 256-bit hash values.

**[0043]** Transaction data of multiple transactions are hashed and stored in a block. For example, hash values of two transactions are provided, and are themselves hashed to provide another hash. This process is repeated until, for all transactions to be stored in a block, a single hash value is provided. This hash value is referred to as a Merkle root hash, and is stored in a header of the block. A change in any of the transactions will result in change in its hash value, and ultimately, a change in the Merkle root hash.

**[0044]** Blocks are added to the blockchain through a consensus protocol. Multiple nodes within the blockchain network participate in the consensus protocol, and perform work to have a block added to the blockchain. Such nodes are referred to as consensus nodes. PBFT, introduced above, is used as a non-limiting example of a consensus protocol. The consensus nodes execute the consensus protocol to add transactions to the blockchain, and update the overall state of the blockchain network.

**[0045]** In further detail, the consensus node generates a block header, hashes all of the transactions in the block, and combines the hash value in pairs to generate further hash values until a single hash value is provided for all transactions in the block (the Merkle root hash). This hash is added to the block header. The consensus node also determines the hash value of the most recent block in the blockchain (i.e., the last block added to the blockchain). The consensus node also adds a nonce value, and a timestamp to the block header.

**[0046]** In general, PBFT provides a practical Byzantine state machine replication that tolerates Byzantine faults (e.g., malfunctioning nodes, malicious nodes). This is achieved in PBFT by assuming that faults will occur (e.g., assuming the existence of independent node failures, and/or manipulated messages sent by consensus nodes). In PBFT, the consensus nodes are provided in a sequence that includes a primary consensus node, and backup consensus nodes. The primary consensus node is periodically changed, Transactions are added to the blockchain by all consensus nodes within the blockchain network reaching an

agreement as to the world state of the blockchain network. In this process, messages are transmitted between consensus nodes, and each consensus nodes proves that a message is received from a specified peer node, and verifies that the message was not modified during transmission.

**[0047]** In PBFT, the consensus protocol is provided in multiple phases with all consensus nodes beginning in the same state. To begin, a client sends a request to the primary consensus node to invoke a service operation (e.g., execute a transaction within the blockchain network). In response to receiving the request, the primary consensus node multicasts the request to the backup consensus nodes. The backup consensus nodes execute the request, and each sends a reply to the client. The client waits until a threshold number of replies are received. In some examples, the client waits for  $f+1$  replies to be received, where  $f$  is the maximum number of faulty consensus nodes that can be tolerated within the blockchain network. The final result is that a sufficient number of consensus nodes come to an agreement on the order of the record that is to be added to the blockchain, and the record is either accepted, or rejected.

**[0048]** In some blockchain networks, cryptography is implemented to maintain privacy of transactions. For example, if two nodes want to keep a transaction private, such that other nodes in the blockchain network cannot discern details of the transaction, the nodes can encrypt the transaction data. An example of cryptography includes, without limitation, symmetric encryption, and asymmetric encryption. Symmetric encryption refers to an encryption process that uses a single key for both encryption (generating ciphertext from plaintext), and decryption (generating plaintext from ciphertext). In symmetric encryption, the same key is available to multiple nodes, so each node can en-/de-crypt transaction data.

**[0049]** Asymmetric encryption uses keys pairs that each include a private key, and a public key, the private key being known only to a respective node, and the public key being known to any or all other nodes in the blockchain network. A node can use the public key of another node to encrypt data, and the encrypted data can be decrypted using other node's private key. For example, and referring again to FIG. 2, Participant A can use Participant B's public key to encrypt data, and send the encrypted data to Participant B. Participant B can use its private key to decrypt the encrypted data (ciphertext) and extract the original data (plaintext). Messages encrypted with a node's public key can only be decrypted using the node's private key.

**[0050]** Asymmetric encryption is used to provide digital signatures, which enables participants in a transaction to confirm other participants in the transaction, as well as the validity of the transaction. For example, a node can digitally sign a message, and another node can confirm that the message was sent by the node based on the digital signature of Participant A. Digital signatures can also be used to ensure that messages are not tampered with in transit. For example, and again referencing FIG. 2, Participant A is to send a message to Participant B. Participant A generates a hash of the message, and then, using its private key, encrypts the hash to provide a digital signature as the encrypted hash. Participant A appends the digital signature to the message, and sends the message with digital signature to Participant B. Participant B decrypts the digital signature using the public key of Participant A, and extracts the hash. Participant B hashes the message and compares the hashes. If the hashes are same, Participant B can confirm that the message was indeed from Participant A, and was not tampered with.

**[0051]** FIG. 3 is a diagram illustrating an example of an environment 300 in accordance with embodiments of this specification. The environment 300 implements trust timestamp services in a blockchain-based centralized ledger system 310. The blockchain-based centralized ledger system 310 adopts a data structure of a blockchain to leverage immutability, reliability, and trustworthiness of data stored on the blockchain. The centralized ledger system 310 can also integrate trust timestamp information from a trust time server 350 that is independent from the centralized ledger system 310 for the data stored on the blockchain, which can greatly enhance credibility, auditability, and legality of the stored data.

**[0052]** In some embodiments, the centralized ledger system 310 can be a cloud computing system including one or more computers interconnected by a network. The centralized ledger system 310 can include any appropriate computing devices. Examples of computing devices include, without limitation, a server, a desktop computer, a laptop computer, a tablet computing device, and a smartphone.

**[0053]** In some examples, the centralized ledger system 310 includes one or more ledger servers 320-1 to 320-n (collectively referred to herein as “320”). Each ledger server 320 can host one or more computer-implemented services for interacting with at least one client, e.g., client 1 or client m. The client can be an individual, a company, an organization, a financial

institution, an insurance company, or any other type of entity. In some cases, a client can be associated with one or more ledger servers. In some cases, a ledger server can be associated with one or more clients.

**[0054]** The ledger server 320 can host a transaction management system to provide a ledger service for a client, e.g., client 1 or client m, and the client can use one or more associated devices, e.g., client device 340-1 or 340-m (collectively referred to herein as “340”), to access the transaction management system to use the ledger service. The client device 340 can include any appropriate computing devices.

**[0055]** The ledger service provided by the ledger server 320 can enable a client to store its data in a transparent, immutable, and cryptographically verifiable blockchain data structure, e.g., a blockchain. Each ledger server, e.g., 320-1 or 320-n, can maintain a respective blockchain, e.g., 322-1 to 322-n (collectively referred to herein as “322”). In some embodiments, each ledger server 320 can perform similar functions to those of a blockchain network node (e.g., the computing device 106 or 108 of FIG. 1 or the computing device 202, 204 or 206 of FIG. 2) in a blockchain network. For example, each ledger server 320 can generate blocks and add the blocks to the blockchain 322. In some embodiments, each ledger server 320 can function as a central trusted authority and does not need to perform consensus processes with other nodes (e.g., other client devices or other ledger servers) to establish trust. For example, each ledger server 320 can perform similar functions to those of a non-consensus node of a blockchain network. In some embodiments, each ledger server 320 can be the single node that creates and/or manages the blockchain 322.

**[0056]** In some embodiments, each client can be associated with a respective blockchain. In some embodiments, one or more clients can be associated with a same blockchain. In some embodiments, a blockchain can be associated with one or more clients.

**[0057]** In some examples, client 1 is an individual, a company, or an organization. The client device 340-1 associated with client 1 can interact with the ledger server 320-1 to obtain a ledger service of the centralized ledger system 310. For example, the client device 340-1 can access the blockchain 322-1 to read and store transaction data associated with client 1 through the ledger server 320-1. The client device 340-1 can include, for example, any suitable computer, module, server, or computing element programmed to perform methods

described herein. In some embodiments, the client device 340-1 can include a user device, such as, a personal computer, a smartphone, a tablet, or other handheld device.

**[0058]** In some examples, client m is an insurance company or a financial institution such as a bank that has a number of individual users. The client device 340-m associated with client m can interact with the ledger server 320-m to provide a ledger service of the centralized ledger system 310 to the individual users of client m. For example, the client device 340-m can access the blockchain 322-m to read and store transaction data associated with client m through the ledger server 320-m. In some cases, a user of client m can request a ledger service of the centralized ledger system 310 through the client device 340-m.

**[0059]** The data stored in a blockchain can be in a log format, including, for example, not only for transaction logs but also other transaction data and block data. Each blockchain stores data in a way that the data is immutable and cannot be altered or deleted. Using cryptography can enable verification that there have been no unintended modification to the stored data. Thus, data recorded on the blockchain are reliable and trustworthy.

**[0060]** The blockchain can include one or more blocks. Each block in the blockchain is linked to a previous block immediately before it in the chain by including a hash of the previous block. Each block also includes a local timestamp, its own hash, and one or more transactions or transaction data. For example, the block can include a block header and a block body. The block header can include the local timestamp, its own hash, and a hash of the previous block. The block body can include payload information such as the one or more transactions or transaction data. The local timestamp indicates a time point or instance when the block is generated and/or added to the blockchain. The local timestamp can be internally provided by the ledger server 320, the centralized ledger system 310, or a central trusted authority associated with the centralized ledger system 310.

**[0061]** In some embodiments, the ledger server 320 sequentially receives a series of transactions associated with a client and then stores the transactions in blocks of a blockchain. In some embodiments, the ledger server 320 can receive one or more transactions, for example, from one or more client devices 340. The received transactions can be stored in a data buffer. The ledger server 320 can generate a block to store the transactions, for example, including transferee and transferor accounts, transaction amounts, or other types of information of the transactions.

**[0062]** In some embodiments, the ledger server 320 can store the transactions in a stream, array, or another data structure (referred to as a transaction storage stream). For example, the transactions can be sequentially stored in the transaction storage stream according to time of occurrence of the transactions. Each transaction can have a respective transaction identifier in the transaction storage stream, for example, according to its time of occurrence. The ledger server 320 can generate blocks to include a number of hashes for the transactions. In some embodiments, the hashes for the transactions can be stored according to the time of occurrence of corresponding transactions, instead of according to values of the hashes. In some embodiments, the hashes for the transactions can be hashes of the transactions or hashes of the respective transaction identifiers of the transactions. A block can include a hash of a previous block immediately before it such that the blocks are anchored with each other to form a blockchain (or a block storage stream). In such a way, the blocks do not store details of the transactions. The details of the transactions can be stored in the transaction storage stream in the ledger server 320 or a separate repository in the centralized ledger system 310.

**[0063]** The ledger server 320 can also provide trust timestamp services to a client. In some embodiments, the ledger server 320 can request trust timestamps from the trust time server 350 for data stored in the ledger server 320, which can enhance credibility, auditability, and legality of the stored data. The trust time server 350 is independent from the centralized ledger system 310. The trust time server 350 can be associated with a third-party trust time authority that provides accurate (or true) time services and can be, for example, globally, acknowledged or trusted by the public, auditing entities (such as companies, institutions, or organizations), and/or legal entities (such as courts or governments). Trust timestamp information provided by the trust time server 350 can be acknowledged or considered as legality without notarization and/or forensic identification. For example, the trust time server 350 can be a UTC (Coordinated Universal Time) / GMT (Greenwich Mean Time) server providing UTC/GMT time services. The trust time server 350 can also be a time server of a trust authority providing standard times for a country or a region.

**[0064]** The centralized ledger system 310 can communicate with the trust time server 350 through a network, e.g., the network 110 of FIG. 1. In response to receiving a timestamp request from a customer, e.g., the ledger server 320, the trust time server 350 can generate a

timestamp indicating a time point when receiving the timestamp request. The trust time server 350 can generate a signature to authenticate the timestamp and the timestamp request (e.g., a textual or imaging copy of the timestamp request). For example, the trust time server 350 can use its private key to sign, thus cryptographically encrypting, the timestamp and the timestamp request. The trust time server 350 can generate a digital timestamp certificate including the timestamp and the associated signature and transmit timestamp information including the timestamp certificate to the customer. The trust time server 350 can provide the trust timestamp service with a cost, e.g., \$ 1 per timestamp request.

**[0065]** In some embodiments, the ledger server 320 transmits to the trust time server 350 a timestamp request for authenticating a time of a block in a blockchain. The timestamp request can include information of the block, e.g., a hash of the block. The time server 350 can generate and transmit timestamp information including the timestamp and associated signature for the block or a hash of the timestamp and associated signature. After receiving the timestamp information from the trust time server 350, the ledger server 320 can store the timestamp information or a hash of the timestamp information into a following block immediately subsequent to the block in the blockchain. In some embodiment, the timestamp information can be stored as a transaction in the following block. A block storing the timestamp information can be referred to be a timestamped block. The timestamped block can be a block that includes only the timestamp information, or a block that also include other transactions in addition to the timestamp information. Timestamped blocks in the blockchain can be anchored or linked to each other in the blockchain.

**[0066]** In some embodiment, the ledger server 320 can periodically transmit timestamp requests for to-be-timestamped blocks in a blockchain to the trust time server 350 with a predetermined triggering time period. For example, the ledger server 320 can include a timer counting a time after transmitting a first timestamp request. When the timer counts the predetermined triggering time period, the ledger server 320 can be triggered to transmit a second timestamp request immediately subsequent to the first timestamp request. The centralized ledger system 310 or the ledger server 320 can provide timestamp services with different costs corresponding to different triggering time periods. The triggering time period can be predetermined by a client (or a user) associated with the blockchain or the ledger

server 320. For example, the client can choose a timestamp service corresponding to a respective cost and a respective triggering time period.

**[0067]** In some embodiments, the ledger server 320 may not transmit timestamp requests to the trust time server 350 periodically. For example, the ledger server 320 may transmit timestamp requests on demand or based on the number of the blocks generated by the ledger server 320. For example, the ledger server 320 may transmit a timestamp request of a block upon receiving instructions from the client, or upon a predetermined number of blocks have been recently added to the blockchain 322.

**[0068]** In some embodiments, the ledger server 320 may generate blocks periodically at a predetermined time period of block generation. The predetermined triggering time period can be the same or different from the time period of block generation. The predetermined triggering time period can be longer than the time period of block generation so that not every block is being timestamped, for example, due to the cost of obtaining the timestamp from the trust time server 350. In some embodiments, the ledger server 320 may not generate blocks periodically. For example, the ledger server 320 may generate blocks on demand or based on the number of the transactions received by the ledger server 320. For example, the ledger server 320 may generate a new block upon receiving a predetermined number of transactions.

**[0069]** In some embodiment, the ledger server 320 can include one or more application programming interfaces (APIs) that is configured to communicate with the trust time server 350. An API can include a set of subroutine definitions, communication protocols, and tools for building software, and defines functionality provided by a program (module, library) and allows abstraction from exactly how this functionality is implemented. Software components interact with each other through the APIs. In some embodiment, the ledger server 320 can include one or more APIs that can implement functionalities of receiving a hash of a to-be-timestamped block as an input for a timestamp request, transmitting the timestamp request to the trust time server 350, and receiving trust timestamp information, e.g., a digital timestamp certificate or a timestamp and associated signature, sent by the trust time server 350.

**[0070]** The ledger server 320 can include one or more APIs that are configured to communicate with a client device 340 associated with a client. The one or more APIs can implement functionalities such as receiving a request for a timestamp service from the client

device 340, listing different timestamp services with different costs and different triggering time periods, receiving a selection among the timestamp services from the client device 340, and transmitting or displaying a corresponding cost with a corresponding triggering time period to the client device 340. In some embodiment, the one or more APIs can also implement functionalities such as receiving a request for verifying or auditing transactions stored on a blockchain associated with the client and transmitting a verification or audition result to the client device 340. As discussed with further details in FIGs. 4A and 4B, the one or more APIs can also implement other functionalities such as receiving transactions or transaction data and client signatures from the client device 340 and transmitting a ledger signature indicating acknowledging the receipt or storage of the transactions or transaction data and/or the client signatures.

**[0071]** In some embodiments, the centralized ledger system 310 includes a centralized server 330. The centralized server 330 can be in communication with the number of ledger servers 320 in the centralized ledger system 310. In some embodiments, the ledger servers 320 communicates with the client devices 340 through the centralized server 330. For example, the centralized server 330 can receive data from a client device 340 and send the data to a ledger server 320 corresponding to (or assigned to) the client device 340.

**[0072]** In some embodiments, the centralized server 330 can maintain a standard time server for the centralized ledger system 310 and can provide internal timestamps (and/or associated signatures) to the ledger servers 320. For example, when a ledger server 320 generates a new block, the ledger server 320 can obtain an internal timestamp (and/or associated signature) from the centralized server 330 and store the internal timestamp (and/or associated signature) in the new block.

**[0073]** In some embodiments, each of the ledger servers 320 communicates with the trust time server 350 through the centralized server 330. For example, the ledger servers 320 can transmit original timestamp requests to the centralized server 330 and the centralized server 330 can transmit the original timestamp requests or centralized server timestamp requests associated with the timestamp requests to the trust time server 350, e.g., through a centralized API in the centralized server 330. The centralized server 330 can provide trust timestamp information obtained from the trust time server 350 to the ledger servers 320. In some other

embodiments, as described above, each of the ledger servers 320 can communicate with the trust time server 350 directly without the centralized server 330.

**[0074]** FIG. 4A is a diagram illustrating an example of a blockchain-based centralized ledger system 400 for implementing a trust timestamp service in a single ledger server associated with a single client in accordance with embodiments of this specification. The blockchain-based centralized ledger system 400 can include a single ledger server 420 dedicated to provide a ledger service to a single client associated with a client device 410. The blockchain-based centralized ledger system 400 can be an example of the centralized ledger system 310 of FIG. 3. For example, the ledger server 420 can be an example of the ledger server 320-1 of FIG. 3. The client device 410 can be an example of the client device 340-1 of FIG. 3. The client uses the client device 410 to access the ledger service provided by the ledger server 420, in the blockchain-based centralized ledger system 400. The ledger server 420 can also provide a trust timestamp service to the client by communicating with a trust time server 430, which can be, for example, the trust time server 350 of FIG. 3.

**[0075]** The ledger server 420 can provide the ledger service and the trust timestamp service exclusively to the client. The ledger server 420 can store transaction data associated with the client in a blockchain exclusively for the client and independent (or separate) from other clients in the centralized ledger system 400. The ledger server 420 can request and store trust timestamp information exclusively for the transaction data associated with the client stored in the blockchain in the ledger server 420. The client can have an administrative right for storing transactions in the blockchain. In some cases, the client can provide to a third party a secondary ledger right that allows the third party to store transactions in the blockchain associated with the client.

**[0076]** In some embodiments, when a transaction (or transaction data) associated with the client is stored in the ledger server 420, the client can use the client device 410 to transmit a client signature to the ledger server 420. The client signature can indicate that the client acknowledges that the transaction has been completed and/or is to be stored in the ledger server 420. Thus, the client cannot repudiate the transaction.

**[0077]** In some embodiments, after receiving and/or storing the transaction (or the transaction data) in the ledger server 420 (e.g., in a blockchain), the ledger server 420 can transmit a ledger signature to the client device 410. The ledger signature can indicate that the

ledger server 420 acknowledges the receipt and/or storage of the transaction. Thus, the ledger server 420 cannot deny storing the transaction.

**[0078]** In some embodiments, the ledger server 420 can transmit to the trust time server 430 a timestamp request for transactions that are associated with the client and stored in the ledger server 420. The trust time server 430 can provide a timestamp and associated signature for the transactions to the ledger server 420. The timestamp signature can include information of the transactions. Thus, the trust time server 430 cannot deny that its endorsement of time of the transactions and the timestamps for the transactions are trustworthy.

**[0079]** In some embodiments, the three respective rights of the three parties (the client, the ledger server, and the trust time server) are independent from each other, which can enhance creditability and auditability of the transaction data stored in the centralized ledger system.

**[0080]** FIG. 4B is a diagram illustrating an example of a blockchain-based centralized ledger system 450 for providing a trust timestamp service to multiple clients by a joint ledger server in accordance with embodiments of this specification. The blockchain-based centralized ledger system 450 can include a single joint ledger server 470 for providing a ledger service to multiple clients, client 1 to client n. The blockchain-based centralized ledger system 450 can be another example of the centralized ledger system 310 of FIG. 3. For example, the joint ledger server 470 can be an example of the ledger server 320 of FIG. 3. Each client, client 1 to client n, can be associated with a respective client device, 460-1 to 460-n. In some embodiments, the client devices 460-1 to 460-n can be examples of the client device 340-1 or 340-m of FIG. 3. Each client can use its respective client device 460 to access the ledger service provided by the ledger server 420, in the blockchain-based centralized ledger system 450. As an example, the clients can include multiple financial institutions such as customer banks.

**[0081]** Each client can use its associated client device to store transactions (or transaction data) in a joint blockchain shared with other clients. Similar to FIG. 4A, each client can transmit a respective client signature to the ledger server 470 and the ledger server 470 can return a corresponding ledger signature to the client. The ledger server 470 can transmit

timestamp requests for the transactions stored in the joint blockchain to the trust time server 430 and receive and store timestamp information for the transactions in the joint blockchain.

**[0082]** FIG. 5 shows an example of a process 500 of managing a centralized trust timestamp blockchain in a blockchain-based centralized ledger system in accordance with embodiments of this specification. The blockchain-based centralized ledger system can be an example of the centralized ledger system 310 of FIG. 3. The centralized ledger system can include a centralized ledger server 520, e.g., the centralized ledger server 330 of FIG. 3. The centralized ledger server 520 can transmit (e.g., periodically) server timestamp requests to a trust time server 510 and maintain a centralized trust timestamp blockchain 530 by storing a series of trust timestamps and associated signatures of the trust time server 510 received from the trust time server 510. The trust time server 510 is associated with a trust time authority and is independent from the centralized ledger system. The trust time server 510 can be, for example, the time server 350 of FIG. 3 or the time server 430 of FIGs. 4A-4B. The centralized ledger server 520 can generate a series of timestamp (TS) blocks, e.g., TS block  $m$  to TS block  $n+1$ , each storing a respective trust timestamp and associated signature of the trust time server 510. The timestamp blocks can be anchored or linked together in the centralized trust timestamp blockchain 530 by, for example, storing a hash of a preceding timestamp block immediately before it in the centralized trust timestamp blockchain 530.

**[0083]** The centralized trust timestamp blockchain 530 can be used to provide timestamp services to multiple blockchains in the centralized ledger system. When the centralized ledger server 520 receives a timestamp request for a block in a blockchain from a ledger server associated with the blockchain, the centralized ledger server 520 can transmit a most recent timestamp and associated signature stored in the centralized trust timestamp blockchain 530 to the ledger server.

**[0084]** As illustrated in FIG. 5, the centralized ledger server 520 can be in communication with multiple ledger servers such as 540-i, 540-j (collectively referred to herein as “540”) that store transaction data in respective blockchains such as 542-i, 542-j (collectively referred to herein as “542”). Each ledger server 540 can be, for example, the ledger server 320 of FIG. 3, 420 of FIG. 4A, or 470 of FIG. 4B. Each blockchain 542 can be, for example, the blockchain 322 of FIG. 3.

**[0085]** Each blockchain 542 can include multiple blocks. Each block has a block identifier and is sequentially added to the blockchain 542 according to the block identifier. Each block in a blockchain is linked to a previous block immediately before it in the chain by including a hash of the previous block. Each block also includes an internal (or local) timestamp, its own hash, and one or more transactions or transaction data. The internal timestamp indicates a time point when the block is generated and added to the blockchain. The internal timestamp can be internally provided by the ledger server that generates the block, the centralized ledger system or a central trusted authority associated with the centralized ledger system. For example, as illustrated in FIG. 5, ledger server 540-i manages a blockchain 542-i that includes blocks i1 to i5. Each of the blocks i1 to i5 has a block identifier i1 to i5 and is sequentially arranged and linked in order in the blockchain 542-i according to the block identifiers. Ledger server 540-j manages a blockchain 542-j that includes blocks j1 to j4. Each of the blocks j1 to j4 has a block identifier j1 to j4 and is sequentially arranged and linked in order in the blockchain 542-j according to the block identifiers.

**[0086]** In the example 500, a ledger server 540 can transmit timestamp requests for blocks in its associated blockchain 542 to the centralized ledger server 520. For example, the ledger server 540 can periodically transmit a timestamp request for a block in the blockchain 542 to the centralized ledger server 520 with a predetermined triggering time period. The predetermined triggering time period can be associated with a corresponding cost for a timestamp service of the centralized ledger system. The predetermined triggering time period can be predetermined by a client (or a user) associated with the blockchain 542 and the ledger server 540. Different ledger servers 540 can have different triggering time periods.

**[0087]** The predetermined triggering time period of the ledger server 540 can be the same or different from a time period of block generation by the ledger server 540. In some embodiments, the triggering time period can be longer than the time period of block generation so that not every block is being timestamped, for example, due to the cost of obtaining the timestamp from the trust time server 510. In some embodiments, the ledger server 540 may not generate blocks periodically. For example, the ledger server 540 may generate blocks on demand or based on the number of the transactions received by the ledger server 540. For example, the ledger server 540 may generate a new block upon receiving a

predetermined number of transactions. In some embodiments, the ledger server 540 may not transmit timestamp requests to the trust time server periodically. For example, the ledger server 540 may transmit timestamp requests on demand or based on the number of the blocks generated by the ledger server 540. For example, the ledger server 540 may transmit a timestamp request of a block upon receiving instructions from the client, or upon a predetermined number of blocks have been recently added to the blockchain.

**[0088]** The ledger server 540 can include a timer counting a time after transmitting a first timestamp request. When the timer counts to the predetermined triggering time period, the ledger server 540 can be triggered to transmit a second timestamp request immediately subsequent to the first timestamp request among timestamp requests transmitted by the ledger server 540 to the centralized ledger server 520. In some cases, the ledger server 540 generates blocks, e.g., according to a predetermined time period of block generation. In some embodiments, the predetermined time period for block generation can be associated with the predetermined triggering time period.

**[0089]** For example, as illustrated in FIG. 5, after transmitting a first timestamp request for the block i2 in the blockchain 542-i, the ledger server 540-i is triggered to transmit a second, immediately subsequent timestamp request for block i5 after a predetermined triggering time period for the ledger server 540-i. Similarly, after transmitting a first timestamp request for the block j2 in the blockchain 542-j, the ledger server 540-j is triggered to transmit a second, immediately subsequent timestamp request for block j4 after a predetermined triggering time period for the ledger server 540-j. The predetermined triggering time period for the ledger server 540-j can be different from the predetermined triggering time period for the ledger server 540-i.

**[0090]** In some embodiments, the centralized ledger server 520 is configured to periodically transmit a centralized server timestamp request to the trust time server 510 with a predetermined triggering time period. The predetermined triggering time period of the centralized ledger server 520 can be shorter than any of the predetermined triggering time periods of the ledger servers 540 in communication with the centralized ledger server 520 (or identical to the shortest predetermined triggering time period of the ledger servers 540), such that the centralized ledger server 520 can accommodate timestamp requests from the ledger servers 540.

[0091] In some embodiments, the server timestamp request is independent from timestamp requests from the ledger servers 540 and includes no information associated with blocks in the blockchains 542. Instead, the server timestamp request includes only information of the centralized trust timestamp blockchain 530. In such a way, the centralized trust timestamp blockchain 530 maintained by the centralized ledger server 520 can be independent from the ledger servers 540 and associated blockchains 542 and can function as an independent time authority for providing trust timestamp services to the ledger servers 540.

[0092] The server timestamp request can include information that uniquely identifies at least one of a most recent timestamp block in the timestamp blockchain 530, a most current timestamp block to be generated in the timestamp blockchain 530, or the server timestamp request itself. In some cases, the server timestamp request includes an identifier of the server timestamp request among the server timestamp requests transmitted to the trust time server 510. In some cases, the server timestamp request includes a block identifier of a most current timestamp block to be generated in the timestamp blockchain 530. In some cases, the server timestamp request includes at least one of a block identifier or a hash of a most recent timestamp block generated in the timestamp blockchain 530.

[0093] In response to receiving a server timestamp request from the centralized ledger server 520, the trust time server 510 can generate trust timestamp information for the server timestamp request. The trust timestamp information includes a trust timestamp indicating a time point receiving the server timestamp request and an associated signature encrypting the trust timestamp and the server timestamp request.

[0094] After receiving the trust timestamp and associated signature from the trust time server 510, the centralized ledger server 520 can generate a new timestamp block to store the trust timestamp and associated signature. The new timestamp block can also include a hash of a preceding timestamp block immediately before the new timestamp block in the centralized timestamp blockchain 530, such that the new timestamp block is added to the centralized timestamp blockchain 530 and linked/anchored to other preceding timestamp blocks in the centralized timestamp blockchain 530.

[0095] When receiving a timestamp request for a block in a blockchain 542 from a ledger server 540 associated with the blockchain 542, the centralized ledger server 520 can transmit

a trust timestamp and associated signature that is stored in the timestamp blockchain 530 and corresponds to the timestamp request to the ledger server 540. The trust timestamp and associated signature can be stored in a most recent timestamp block generated in the timestamp blockchain when the timestamp request is received. In some embodiments, the centralized ledger server 520 can generate an internal (or local) timestamp for the timestamp block storing the trust timestamp and associated signature according to the centralized trust timestamp blockchain 530 and generate an internal signature encrypting the internal timestamp and the timestamp block. The centralized ledger server 520 can also transmit the internal timestamp and the internal signature to the ledger server 540, together with the trust timestamp and associated signature obtained from the trust time server 510.

**[0096]** The ledger server 540 can then store the timestamp and associated signature as a transaction in the blockchain 542. In some cases, the block to be timestamped can be the most recent block generated in the blockchain 542 when the timestamp request is transmitted. The timestamp request can include a hash of the most recent generated block or other information uniquely identifying the most recent generated block. After receiving the corresponding timestamp and associated signature for the most recent generated block, the ledger server 540 can store the corresponding timestamp and associated signature and/or a hash thereof as a transaction in a block immediately subsequent to the most recent generated block in the blockchain 542. The block immediately subsequent to the most recent generated block can also store the hash of the most recent generated block.

**[0097]** In some cases, the block to be timestamped can be the most current block to be generated in the blockchain 542 when the timestamp request is transmitted. The timestamp request can include a hash of transaction data to be stored in the most current block to be generated. After receiving the corresponding timestamp and associated signature for the most recent generated block, the ledger server 540 can generate the block storing the corresponding timestamp and associated signature as a transaction, together with the transaction data.

**[0098]** For example, as illustrated in FIG. 5, during a first predetermined triggering time period of the centralized ledger server 520 that is after generating the timestamp block  $m+2$  and before generating the timestamp block  $m+3$ , the most recent generated timestamp block in the centralized trust timestamp blockchain 530 is timestamp block  $m+2$ . During the first

predetermined triggering time period, the centralized ledger server 520 receives a timestamp request for block i2 in the blockchain 542-i from the ledger server 540-i and a timestamp request for block j2 in the blockchain 542-j from the ledger server 540-j. The blocks i2 and j2 can be the most current blocks to be generated in the blockchain 542-i and 542-j, respectively. The timestamp requests can respectively include hashes of transaction data to be stored in the block i2 and the block j2. The centralized ledger server 520 can transmit the trust timestamp and associated signature stored in the most recent generated timestamp block m+2 to the ledger server 540-i and the ledger server 540-j. The ledger server 540-i can generate the block i2 that stores the trust timestamp and associated signature stored in the timestamp block m+2 as a transaction and the transaction data for the block i2. The ledger server 540-j can generate the block j2 that stores the trust timestamp and associated signature stored in the timestamp block m+2 as a transaction and the transaction data for the block j2.

**[0099]** As illustrated in FIG. 5, during a second predetermined triggering time period of the centralized ledger server 520 that is after generating the timestamp block n and before generating the timestamp block n+1, the most recent generated timestamp block in the centralized trust timestamp blockchain530 is timestamp block n. During the second predetermined triggering time period, the centralized ledger server 520 receives a timestamp request for block i5 in the blockchain 542-i from the ledger server 540-i and a timestamp request for block j4 in the blockchain 542-j from the ledger server 540-j. The blocks i5 and j4 can be the most current blocks to be generated in the blockchain 542-i and 542-j, respectively. The timestamp requests can respectively include hashes of transaction data to be stored in the block i5 and the block j4. The centralized ledger server 520 can transmit the trust timestamp and associated signature stored in the most recent generated timestamp block n to the ledger server 540-i and the ledger server 540-j. The ledger server 540-i can generate the block i5 that stores the trust timestamp and associated signature stored in the timestamp block n as a transaction and the transaction data for the block i5. The ledger server 540-j can generate the block j4 that stores the trust timestamp and associated signature stored in the timestamp block n as a transaction and the transaction data for the block j4.

**[0100]** As noted above, the predetermined triggering time period of the centralized ledger server 520 can be shorter than any of the predetermined triggering time periods of the ledger servers 540, such that the centralized trust timestamp blockchain530 can accommodate

timestamp requests from the ledger servers 540. In some embodiments, the second predetermined triggering time period can be not immediately subsequent to the first predetermined triggering time period, and the timestamp block  $n$  can be separated from the timestamp block  $m+2$  by one or more other timestamp blocks in the centralized trust timestamp blockchain 530. The timestamp requests for block  $i2$  and  $i5$  can be immediately subsequent among the timestamp requests transmitted from the ledger server 540- $i$  to the centralized ledger server 520. The timestamp requests for block  $j2$  and  $j4$  can be immediately subsequent among the timestamp requests transmitted from the ledger server 540- $j$  to the centralized ledger server 520.

**[0101]** In some embodiments, the centralized timestamp blockchain 530 includes timestamp blocks each storing respective trust timestamp and associated signature of the trust time server 510 and the ledger servers 540 store trust timestamps and associated signatures obtained from the centralized timestamp blockchain 530 for blocks of blockchains associated with the ledger servers 540, orderliness and authenticity of transaction data stored in the blocks of the blockchains can be enhanced. For example, as illustrated in FIG. 5, for blockchain 542- $i$ , block  $i2$  includes a first trust timestamp from the timestamp block  $m+2$ , which can guarantee one or more blocks preceding block  $i2$ , e.g., block  $i1$ , are generated before the first trust timestamp. Block  $i5$  includes a second trust timestamp from the timestamp block  $n$ , which can guarantee one or more blocks preceding block  $i5$ , e.g., blocks  $i1$  to  $i4$ , are generated before the second trust timestamp. Moreover, the timestamp block  $m+2$  and timestamp block  $n$  are sequentially anchored together in the centralized timestamp blockchain 530 and the blocks  $i2$  to  $i5$  are sequentially anchored together in the blockchain 542- $i$ . Thus, the centralized timestamp blockchain 530 can guarantee that blocks between the blocks  $i2$  and  $i5$ , i.e., blocks  $i3$  and  $i4$ , are generated at time points between the first trust timestamp in the block  $i2$  and the second trust timestamp in the block  $i5$ .

**[0102]** Similarly, for blockchain 542- $j$ , block  $j2$  includes the first trust timestamp from the timestamp block  $m+2$ , which can guarantee one or more blocks preceding block  $j2$ , e.g., block  $j1$ , are generated before the first trust timestamp. Block  $j4$  includes the second trust timestamp from the timestamp block  $n$ , which can guarantee one or more blocks preceding block  $j4$ , e.g., blocks  $j1$  to  $j3$ , are generated before the second trust timestamp. Moreover, the timestamp block  $m+2$  and timestamp block  $n$  are sequentially anchored together in the

centralized timestamp blockchain 530 and the blocks j2 to j4 are sequentially anchored together in the blockchain 542-j. Thus, the centralized timestamp blockchain 530 can guarantee that one or more blocks between the blocks j2 and j4, i.e., block j3, are generated at time points between the first trust timestamp in the block j2 and the second trust timestamp in the block j4.

**[0103]** FIG. 6 is a flowchart illustrating an example of a process 600 for implementation of timestamp services that can be executed in accordance with embodiments of this specification. For convenience, the process 600 will be described as being performed by a system of one or more computers, located in one or more locations, and programmed appropriately in accordance with this specification. For example, a centralized ledger server in a blockchain-based centralized ledger system can perform the process 600. The centralized ledger system can be an example of the centralized ledger system 310 of FIG. 3. The centralized ledger server can be an example of the centralized ledger server 330 of FIG. 3 or 520 of FIG. 5.

**[0104]** At 602, timestamps and associated signatures are received from a trust time server. The trust time server is associated with a trust time authority independent from the centralized ledger system. The trust time server can be, for example, the time server 350 of FIG. 3, the time server 430 of FIGs. 4A-4B, or the time server 510 of FIG. 5.

**[0105]** In some embodiments, the centralized ledger server is configured to periodically transmit server timestamp requests to the trust time server with a predetermined triggering time period. Transmitting a server timestamp request can be in response to determining that the predetermined triggering time period has passed after a preceding server timestamp request immediately before the server timestamp request is transmitted.

**[0106]** In some embodiments, the server timestamp request includes only information of the centralized trust timestamp blockchain. In such a way, the centralized trust timestamp blockchain maintained by the centralized ledger server can function as an independent time authority for providing trust timestamp services. In some cases, the server timestamp request includes an identifier of the server timestamp request among the server timestamp requests transmitted to the trust time server. In some cases, the server timestamp request includes a block identifier of a most current timestamp block to be generated in the timestamp blockchain. In some cases, the server timestamp request includes at least one of a block

identifier or a hash of a most recent timestamp block generated in the centralized trust timestamp blockchain.

**[0107]** In response to receiving a server timestamp request from the centralized ledger server, the trust time server can generate trust timestamp information for the server timestamp request. The trust timestamp information can include a trust timestamp indicating a time point receiving the centralized server timestamp request and an associated signature encrypting the trust timestamp and the centralized server timestamp request. The trust time server can then transmit the trust timestamp and associated signature to the centralized ledger server.

**[0108]** At 604, the timestamps and associated signatures of the trust time server are stored in a centralized trust timestamp blockchain. The timestamp blockchain can be, for example, the timestamp blockchain 530 of FIG. 5. In response to receiving an individual timestamp and associated signature from the trust time server, the centralized ledger server can generate a new timestamp block that stores the trust timestamp and associated signature. The new timestamp block can also include a hash of a preceding timestamp block immediately before the new timestamp block in the centralized timestamp blockchain, such that the new timestamp block can be added to the centralized timestamp blockchain and linked/anchored to other preceding timestamp blocks in the centralized timestamp blockchain. The centralized ledger server can maintain the centralized timestamp blockchain by sequentially store trust timestamps and associated signatures of the trust time server in timestamp blocks according to a chronological order of receiving the timestamps and associated signatures from the trust time server. In some embodiments, the centralized ledger server can generate an internal (or local) timestamp for a new timestamp block and an internal (or local) signature encrypting the internal timestamp and the new timestamp block.

**[0109]** At 606, a first timestamp request for a first block of a first blockchain is received from a first ledger server by the centralized ledger server. The centralized ledger system can include multiple ledger servers maintaining multiple blockchains. The centralized ledger server can be in communication with the multiple ledger servers. Each ledger server can be, for example, the ledger server 320 of FIG. 3, 420 of FIG. 4A, 470 of FIG. 4B, or 540 of FIG. 5. Each blockchain can be the blockchain 322 of FIG. 3 or the blockchain 542 of FIG. 5. Each of the blockchains can include multiple blocks storing transaction data.

**[0110]** In some embodiments, a ledger server is configured to periodically transmit timestamp requests for blocks in its corresponding blockchain to the centralized ledger server at a predetermined triggering time period. The predetermined triggering time period can be associated with a corresponding cost for a timestamp service of the centralized ledger system. The predetermined triggering time period can be predetermined by a client associated with the centralized ledger system. Different ledger servers can have different predetermined triggering time period.

**[0111]** At 608, a first timestamp and associated signature that is stored in the timestamp blockchain and corresponds to the first timestamp request is transmitted to the first ledger server by the centralized ledger server. The first timestamp and associated signature can be in a most recent timestamp block generated in the centralized trust timestamp blockchain when the first timestamp request is received by the centralized ledger server. In some embodiments, the centralized ledger server can also transmit to the first ledger server an internal timestamp and an internal signature for the most recent timestamp block storing the first timestamp and associated signature.

**[0112]** After receiving the first timestamp and associated signature from the centralized ledger server, the first ledger server can store the first timestamp and associated signature in the first blockchain. In some embodiments, the first block is a most current block to be generated in the first blockchain when the first ledger server transmits the first timestamp request to the centralized ledger server. The first timestamp request can include a hash of transaction data to be stored in the first block. After receiving the first timestamp and associated signature, the first ledger server can generate the most current block including the first timestamp and associated signature as a transaction and the transaction data for the first block.

**[0113]** In some embodiments, the first block is a most recent generated block in the first blockchain when the first ledger server transmits the first timestamp request to the centralized ledger server. The first timestamp request can include a hash of the first block. After receiving the first timestamp and associated signature, the ledger server can store the first timestamp and associated signature as a transaction in a second block immediately subsequent to the first block in the first blockchain. The second block can also store the hash of the first block. In some embodiments, the second block can store the first timestamp and

associated signature together with transaction data for the second block. In some embodiments, the second block can be a block whose block body exclusively includes the first timestamp and associated signature without storing other transaction data.

**[0114]** FIG. 7 depicts examples of modules of an apparatus 700 in accordance with embodiments of this specification. The apparatus 700 can be an example of an embodiment of a blockchain-based centralized ledger system configured to provide ledger services and/or trust timestamp services for transaction data stored in the centralized ledger system. The apparatus 700 can correspond to the embodiments described above, and the apparatus 700 includes the following: a first receiving module 702 that receives timestamps and associated signatures from a trust time server that is associated with a trust time authority and independent from the blockchain-based centralized ledger system; a storing module 704 that stores the timestamps and associated signatures of the trust time server in a centralized timestamp blockchain, the centralized trust timestamp blockchain including a plurality of timestamp blocks storing trust timestamp information of the trust time server for the blockchain-based centralized ledger system that stores data in a plurality of blockchains, each of the plurality of blockchains including a plurality of blocks storing transaction data; a second receiving module 706 that receives a first timestamp request for a first block of a first blockchain from a first ledger server; and a transmitting module 708 that transmits a first timestamp and associated signature that is stored in the timestamp blockchain and corresponds to the first timestamp request to the first ledger server.

**[0115]** In an optional embodiment, the first block is a most current block to be generated in the first blockchain when the first timestamp request is transmitted by the first ledger server, and the first timestamp request includes a hash of transaction data to be stored in the first block.

**[0116]** In an optional embodiment, the first ledger server includes a generating module that generates the first block storing the first timestamp and the associated signature as a transaction and the transaction data by the first ledger server.

**[0117]** In an optional embodiment, the first block is a most recent block generated in the first blockchain when the first timestamp request is transmitted by the first ledger server, and the first timestamp request includes a hash of the first block.

**[0118]** In an optional embodiment, the first ledger server includes a storing module that stores the first timestamp and the associated signature as a transaction in a second block immediately subsequent to the first block in the first blockchain by the first ledger server. The second block can store the hash of the first block.

**[0119]** In an optional embodiment, the apparatus 700 further includes a second transmitting module that periodically transmits server timestamp requests to the trust time server.

**[0120]** In an optional embodiment, each of the server timestamp requests includes at least one of: an identifier of the server timestamp request among the server timestamp requests, an identifier of a most current timestamp block to be generated in the centralized trust timestamp blockchain, or at least one of an identifier or a hash of a most recent timestamp block generated in the centralized trust timestamp blockchain.

**[0121]** In an optional embodiment, the storing module 704 is configured to: in response to receiving a particular timestamp and associated signature from the trust time server, generate a particular timestamp block in the centralized trust timestamp blockchain, the particular timestamp block including the particular timestamp and the associated signature as a transaction.

**[0122]** In an optional embodiment, the first timestamp and the associated signature are stored in a most recent timestamp block generated in the centralized trust timestamp blockchain when the first timestamp request is received.

**[0123]** In an optional embodiment, the apparatus 700 includes a generating module that generates an internal timestamp signature for the most recent timestamp block storing the first timestamp and associated signature and an internal signature encrypting the internal timestamp and the most recent timestamp block. The transmitting module 708 can be configured to transmit the internal timestamp and the internal signature, together with the first timestamp and associated signature, to the first ledger server.

**[0124]** In an optional embodiment, the first ledger server is configured to periodically transmit timestamp requests for blocks in the first blockchain to the centralized ledger server.

**[0125]** In an optional embodiment, the second receiving module 706 is configured to receive a second timestamp request from a second ledger server associated with a second blockchain different from the first blockchain, and the transmitting module 708 is configured

to transmit a second timestamp and associated signature that is stored in the timestamp blockchain and corresponds to the second timestamp request to the second ledger server.

**[0126]** In an optional embodiment, each of ledger servers in the centralized ledger system is configured to maintain a plurality of blockchains each including a plurality of blocks storing transaction data.

**[0127]** In an optional embodiment, the apparatus 700 further includes a maintaining module that maintains the centralized trust timestamp blockchain by sequentially storing timestamps and associated signatures in respective timestamp blocks according to a chronological order of receiving the timestamps and associated signatures from the trust time server. The blockchain-based centralized ledger system maintains the plurality of blockchains by a plurality corresponding ledger servers.

**[0128]** The system, apparatus, module, or unit illustrated in the previous embodiments can be implemented by using a computer chip or an entity, or can be implemented by using a product having a certain function. A typical embodiment device is a computer (and the computer can be a personal computer), a laptop computer, a cellular phone, a camera phone, a smartphone, a personal digital assistant, a media player, a navigation device, an email receiving and sending device, a game console, a tablet computer, a wearable device, or any combination of these devices.

**[0129]** For an embodiment process of functions and roles of each module in the apparatus, references can be made to an embodiment process of corresponding steps in the previous method. Details are omitted here for simplicity.

**[0130]** Because an apparatus embodiment basically corresponds to a method embodiment, for related parts, references can be made to related descriptions in the method embodiment. The previously described apparatus embodiment is merely an example. The modules described as separate parts may or may not be physically separate, and parts displayed as modules may or may not be physical modules, may be located in one position, or may be distributed on a number of network modules. Some or all of the modules can be selected based on actual demands to achieve the objectives of the solutions of the specification. A person of ordinary skill in the art can understand and implement the embodiments of the present application without creative efforts.

**[0131]** Referring again to FIG. 7, it can be interpreted as illustrating an internal functional module and a structure of a blockchain-based centralized ledger implementation apparatus. The blockchain-based centralized ledger implementation apparatus can be an example of a centralized ledger system configured to provide ledger services and trust timestamp services for transaction data stored in the centralized ledger system. An execution body in essence can be an electronic device, and the electronic device includes the following: one or more processors; and one or more computer-readable memories configured to store an executable instruction of the one or more processors. In some embodiments, the one or more computer-readable memories are coupled to the one or more processors and have programming instructions stored thereon that are executable by the one or more processors to perform algorithms, methods, functions, processes, flows, and procedures, as described in this specification.

**[0132]** Described embodiments of the subject matter can include one or more features, alone or in combination. For example, in a first embodiment, a method includes: receiving timestamps and associated signatures from a trust time server by a centralized ledger server in the blockchain-based centralized ledger system, the trust time server being associated with a trust time authority and independent from the blockchain-based centralized ledger system; storing the timestamps and the associated signatures of the trust time server in the centralized trust timestamp blockchain by the centralized ledger server, the centralized trust timestamp blockchain including a plurality of timestamp blocks storing trust timestamp information of the trust time server for a blockchain-based centralized ledger system that stores data in a plurality of blockchains, each of the plurality of blockchains including a plurality of blocks storing transaction data; receiving a first timestamp request for a first block of a first blockchain from a first ledger server associated with the first blockchain by the centralized ledger server; and transmitting a first timestamp and associated signature that is stored in the timestamp blockchain and corresponds to the first timestamp request to the first ledger server by the centralized ledger server.

**[0133]** The foregoing and other described embodiments can each, optionally, include one or more of the following features:

**[0134]** A first feature, combinable with any of the following features, specifies that the first block is a most current block to be generated in the first blockchain when the first

timestamp request is transmitted by the first ledger server, and the first timestamp request includes a hash of transaction data to be stored in the first block.

**[0135]** A second feature, combinable with any of the previous or following features, specifies that the first ledger server is configured to generate the first block storing the first timestamp and the associated signature as a transaction and the transaction data.

**[0136]** A third feature, combinable with any of the previous or following features, specifies that the first block is a most recent block generated in the first blockchain when the first timestamp request is transmitted by the first ledger server, and the first timestamp request includes a hash of the first block.

**[0137]** A fourth feature, combinable with any of the previous or following features, specifies that the first ledger server is configured to store the first timestamp and the associated signature as a transaction in a second block immediately subsequent to the first block in the first blockchain, and the second block stores the hash of the first block.

**[0138]** A fifth feature, combinable with any of the previous or following features, further includes: periodically transmitting server timestamp requests to the trust time server by the centralized ledger server.

**[0139]** A sixth feature, combinable with any of the previous or following features, specifies that each of the server timestamp requests comprises at least one of: an identifier of the server timestamp request among the server timestamp requests, an identifier of a most current timestamp block to be generated in the centralized trust timestamp blockchain, or at least one of an identifier or a hash of a most recent timestamp block generated in the centralized trust timestamp blockchain.

**[0140]** A seventh feature, combinable with any of the previous or following features, specifies that storing the timestamps and the associated signatures of the trust time server in a centralized trust timestamp blockchain includes: in response to receiving a particular timestamp and associated signature from the trust time server, generating a particular timestamp block in the centralized trust timestamp blockchain, the particular timestamp block including the particular timestamp and the associated signature as a transaction.

**[0141]** An eight feature, combinable with any of the previous or following features, specifies that the first timestamp and the associated signature are stored in a most recent

timestamp block generated in the centralized trust timestamp blockchain when the first timestamp request is received.

**[0142]** A ninth feature, combinable with any of the previous or following features, further includes: generating an internal timestamp signature for the most recent timestamp block storing the first timestamp and associated signature by the centralized ledger server; generating an internal signature encrypting the internal timestamp and the most recent timestamp block by the centralized ledger server; and transmitting the internal timestamp and the internal signature, together with the first timestamp and associated signature, to the first ledger server by the centralized ledger server.

**[0143]** A tenth feature, combinable with any of the previous or following features, specifies that the first ledger server is configured to periodically transmit timestamp requests for blocks in the first blockchain to the centralized ledger server.

**[0144]** An eleventh feature, combinable with any of the previous or following features, further includes: receiving a second timestamp request from a second ledger server associated with a second blockchain different from the first blockchain by the centralized ledger server; and transmitting a second timestamp and associated signature that is stored in the timestamp blockchain and corresponds to the second timestamp request to the second ledger server by the centralized ledger server.

**[0145]** A twelfth feature, combinable with any of the previous or following features, further includes: maintaining a plurality of blockchains by corresponding ledger servers in the centralized ledger system and maintaining the centralized trust timestamp blockchain by the centralized ledger server, where the centralized ledger server is configured to sequentially store timestamps and associated signatures of the trust time server in respective timestamp blocks according to a chronological order of receiving the timestamps and associated signatures from the trust time server.

**[0146]** Embodiments of the subject matter and the actions and operations described in this specification can be implemented in digital electronic circuitry, in tangibly-embodied computer software or firmware, in computer hardware, including the structures disclosed in this specification and their structural equivalents, or in combinations of one or more of them. Embodiments of the subject matter described in this specification can be implemented as one or more computer programs, e.g., one or more modules of computer program instructions,

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

**[0147]** A computer program, which may also be referred to or described as a program, software, a software application, an app, a module, a software module, an engine, a script, or code, can be written in any form of programming language, including compiled or interpreted languages, or declarative or procedural languages; and it can be deployed in any form, including as a stand-alone program or as a module, component, engine, subroutine, or other unit suitable for executing in a computing environment, which environment may include one or more computers interconnected by a data communication network in one or more locations.

**[0148]** A computer program may, but need not, correspond to a file in a file system. A computer program can be stored in a portion of a file that holds other programs or data, e.g., one or more scripts stored in a markup language document, in a single file dedicated to the program in question, or in multiple coordinated files, e.g., files that store one or more modules, sub programs, or portions of code.

**[0149]** Processors for execution of a computer program include, by way of example, both general- and special-purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive the instructions of the computer program for execution as well as data from a non-transitory computer-readable medium coupled to the processor.

**[0150]** The term “data processing apparatus” encompasses all kinds of apparatuses, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. Data processing apparatus can include special-purpose logic circuitry, e.g., an FPGA (field programmable gate array), an ASIC (application specific integrated circuit), or a GPU (graphics processing unit). The apparatus can also include, in addition to hardware, code that creates an execution environment for computer programs, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them.

**[0151]** The processes and logic flows described in this specification can be performed by one or more computers or processors executing one or more computer programs to perform operations by operating on input data and generating output. The processes and logic flows can also be performed by special-purpose logic circuitry, e.g., an FPGA, an ASIC, or a GPU, or by a combination of special-purpose logic circuitry and one or more programmed computers.

**[0152]** Computers suitable for the execution of a computer program can be based on general or special-purpose microprocessors or both, or any other kind of central processing unit. Generally, a central processing unit will receive instructions and data from a read only memory or a random access memory or both. Elements of a computer can include a central processing unit for executing instructions and one or more memory devices for storing instructions and data. The central processing unit and the memory can be supplemented by, or incorporated in, special-purpose logic circuitry.

**[0153]** Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to one or more storage devices. The storage devices can be, for example, magnetic, magneto optical, or optical disks, solid state drives, or any other type of non-transitory, computer-readable media. However, a computer need not have such devices. Thus, a computer may be coupled to one or more storage devices, such as, one or more memories, that are local and/or remote. For example, a computer can include one or more local memories that are integral components of the computer, or the computer can be coupled to one or more remote memories that are in a cloud network. Moreover, a computer can be embedded in another device, e.g., a mobile telephone, a personal digital assistant (PDA), a

mobile audio or video player, a game console, a Global Positioning System (GPS) receiver, or a portable storage device, e.g., a universal serial bus (USB) flash drive, to name just a few.

**[0154]** Components can be “coupled to” each other by being commutatively such as electrically or optically connected to one another, either directly or via one or more intermediate components. Components can also be “coupled to” each other if one of the components is integrated into the other. For example, a storage component that is integrated into a processor (e.g., an L2 cache component) is “coupled to” the processor.

**[0155]** To provide for interaction with a user, embodiments of the subject matter described in this specification can be implemented on, or configured to communicate with, a computer having a display device, e.g., a LCD (liquid crystal display) monitor, for displaying information to the user, and an input device by which the user can provide input to the computer, e.g., a keyboard and a pointing device, e.g., a mouse, a trackball or touchpad. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback, e.g., visual feedback, auditory feedback, or tactile feedback; and input from the user can be received in any form, including acoustic, speech, or tactile input. In addition, a computer can interact with a user by sending documents to and receiving documents from a device that is used by the user; for example, by sending web pages to a web browser on a user’s device in response to requests received from the web browser, or by interacting with an app running on a user device, e.g., a smartphone or electronic tablet. Also, a computer can interact with a user by sending text messages or other forms of message to a personal device, e.g., a smartphone that is running a messaging application, and receiving responsive messages from the user in return.

**[0156]** This specification uses the term “configured to” in connection with systems, apparatus, and computer program components. For a system of one or more computers to be configured to perform particular operations or actions means that the system has installed on it software, firmware, hardware, or a combination of them that in operation cause the system to perform the operations or actions. For one or more computer programs to be configured to perform particular operations or actions means that the one or more programs include instructions that, when executed by data processing apparatus, cause the apparatus to perform the operations or actions. For special-purpose logic circuitry to be configured to perform

particular operations or actions means that the circuitry has electronic logic that performs the operations or actions.

**[0157]** While this specification contains many specific embodiment details, these should not be construed as limitations on the scope of what is being claimed, which is defined by the claims themselves, but rather as descriptions of features that may be specific to particular embodiments. Certain features that are described in this specification in the context of separate embodiments can also be realized in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also be realized in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially be claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claim may be directed to a subcombination or variation of a subcombination.

**[0158]** Similarly, while operations are depicted in the drawings and recited in the claims in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. In certain circumstances, multitasking and parallel processing may be advantageous. Moreover, the separation of various system modules and components in the embodiments described above should not be understood as requiring such separation in all embodiments, and it should be understood that the described program components and systems can generally be integrated together in a single software product or packaged into multiple software products.

**[0159]** Particular embodiments of the subject matter have been described. Other embodiments are within the scope of the following claims. For example, the actions recited in the claims can be performed in a different order and still achieve desirable results. As one example, the processes depicted in the accompanying figures do not necessarily require the particular order shown, or sequential order, to achieve desirable results. In some cases, multitasking and parallel processing may be advantageous.

## CLAIMS

1. A computer-implemented method for managing a blockchain-based centralized ledger system, the computer-implemented method comprising:

receiving a trust timestamp and an associated signature from a trust time server by a centralized ledger server in the centralized ledger system, the trust time server being associated with a trust time authority and independent from the centralized ledger system, the centralized ledger system storing data in a plurality of data blockchains, each of the plurality of data blockchains comprising a plurality of data blocks storing transaction data;

storing the trust timestamp and the associated signature from the trust time server in a centralized trust timestamp blockchain by the centralized ledger server, the centralized trust timestamp blockchain comprising a plurality of timestamp blocks storing trust timestamp information of the trust time server for the centralized ledger system; and

maintaining the centralized trust timestamp blockchain by the centralized ledger server, wherein the centralized ledger server is configured to sequentially store trust timestamps and associated signatures received from the trust time server in respective timestamp blocks according to a chronological order of receiving the trust timestamps and associated signatures from the trust time server.

2. The computer-implemented method of claim 1, further comprising:

periodically transmitting a series of trust timestamp requests to the trust time server by the centralized ledger server.

3. The computer-implemented method of claim 2, wherein each trust timestamp request of the series of trust timestamp requests comprises at least one of:

an identifier of the trust timestamp request among the series of trust timestamp requests,  
an identifier of a most current timestamp block to be generated in the centralized trust timestamp blockchain, or

at least one of an identifier or a hash of a most recent timestamp block generated in the centralized trust timestamp blockchain.

4. The computer-implemented method claim 1, wherein storing the trust timestamp and the associated signature from the trust time server in the centralized trust timestamp blockchain comprises:

generating a first timestamp block in the centralized trust timestamp blockchain to store the trust timestamp and the associated signature as a transaction, the first timestamp block being a most recent timestamp block generated in the centralized trust timestamp blockchain,

wherein the first timestamp block stores a hash of a second timestamp block immediately preceding the first timestamp block in the centralized trust timestamp blockchain.

5. The computer-implemented method of claim 1, wherein the centralized trust timestamp blockchain is different from each of the plurality of data blockchains and includes no data block.

6. The computer-implemented method of claim 1, wherein each of the plurality of timestamp blocks in the centralized trust timestamp blockchain is different from each of the data blocks in the plurality of data blockchains and stores no transaction data.

7. The computer-implemented method of claim 1, further comprising:

receiving a timestamp request for a data block of a data blockchain from a ledger server associated with the data blockchain by the centralized ledger server; and

transmitting a timestamp and an associated signature that are stored in the centralized trust timestamp blockchain and corresponds to the timestamp request to the ledger server by the centralized ledger server.

8. The computer-implemented method of claim 7, wherein the data block is a most recent data block generated in the data blockchain when the timestamp request is transmitted by the ledger server, and

wherein the timestamp request comprises a hash of the data block.

9. The computer-implemented method of claim 7, wherein the timestamp and the associated signature are stored in a most recent timestamp block generated in the centralized trust timestamp blockchain when the timestamp request is received by the centralized ledger server.

10. The computer-implemented method of claim 9, further comprising:  
generating an internal timestamp for the most recent timestamp block storing the timestamp and associated signature by the centralized ledger server;  
generating an internal signature encrypting the internal timestamp and the most recent timestamp block by the centralized ledger server; and  
transmitting the internal timestamp and the internal signature, together with the timestamp and associated signature, to the ledger server by the centralized ledger server.
11. A non-transitory, computer-readable medium storing one or more instructions executable by a computer-implemented system to perform operations for managing a blockchain-based centralized ledger system, the operations comprising:  
receiving a trust timestamp and an associated signature from a trust time server, the trust time server being associated with a trust time authority and independent from the centralized ledger system, the centralized ledger system storing data in a plurality of data blockchains, each of the plurality of data blockchains comprising a plurality of data blocks storing transaction data;  
storing the trust timestamp and the associated signature from the trust time server in a centralized trust timestamp blockchain, the centralized trust timestamp blockchain comprising a plurality of timestamp blocks storing trust timestamp information of the trust time server for the centralized ledger system; and  
maintaining the centralized trust timestamp blockchain in the centralized ledger system by sequentially storing trust timestamps and associated signatures received from the trust time server in respective timestamp blocks according to a chronological order of receiving the trust timestamps and associated signatures from the trust time server.
12. The non-transitory, computer-readable medium of claim 11, wherein the operations further comprise:  
periodically transmitting a series of trust timestamp requests to the trust time server.
13. The non-transitory, computer-readable medium of claim 12, wherein each trust timestamp request of the series of trust timestamp requests comprises at least one of:

an identifier of the trust timestamp request among the series of trust timestamp requests,  
an identifier of a most current timestamp block to be generated in the centralized trust  
timestamp blockchain, or

at least one of an identifier or a hash of a most recent timestamp block generated in the  
centralized trust timestamp blockchain.

14. The non-transitory, computer-readable medium of claim 11, wherein storing the trust  
timestamp and the associated signature from the trust time server in the centralized trust  
timestamp blockchain comprises:

generating a first timestamp block in the centralized trust timestamp blockchain to store  
the trust timestamp and the associated signature as a transaction, the first timestamp block being  
a most recent timestamp block generated in the centralized trust timestamp blockchain,

wherein the first timestamp block stores a hash of a second timestamp block immediately  
preceding the first timestamp block in the centralized trust timestamp blockchain.

15. The non-transitory, computer-readable medium of claim 11, wherein the centralized trust  
timestamp blockchain is different from each of the plurality of data blockchains and includes no  
data block.

16. The non-transitory, computer-readable medium of claim 11, wherein each of the plurality  
of timestamp blocks in the centralized trust timestamp blockchain is different from each of the  
data blocks in the plurality of data blockchains and stores no transaction data.

17. The non-transitory, computer-readable medium of claim 11, wherein the operations  
further comprise:

receiving a timestamp request for a data block of a data blockchain from a ledger server  
associated with the data blockchain; and

transmitting a timestamp and associated signature that are stored in the centralized trust  
timestamp blockchain and corresponds to the timestamp request to the ledger server.

18. The non-transitory, computer-readable medium of claim 17, wherein the data block is a most recent data block generated in the data blockchain when the timestamp request is transmitted by the ledger server, and

wherein the timestamp request comprises a hash of the data block.

19. The non-transitory, computer-readable medium of claim 17, wherein the timestamp and the associated signature are stored in a most recent timestamp block generated in the centralized trust timestamp blockchain when the timestamp request is received, and

wherein the operations further comprise:

generating an internal timestamp for the most recent timestamp block storing the timestamp and associated signature;

generating an internal signature encrypting the internal timestamp and the most recent timestamp block; and

transmitting the internal timestamp and the internal signature, together with the timestamp and associated signature, to the ledger server.

20. A computer-implemented system, comprising:

one or more processors; and

one or more non-transitory machine readable storage medium coupled to the one or more processors and having machine-executable instructions stored thereon that, when executed by the one or more processors, cause the one or more processors to perform operations for managing a blockchain-based centralized ledger system, the operations comprising:

receiving a trust timestamp and an associated signature from a trust time server, the trust time server being associated with a trust time authority and independent from the centralized ledger system, the centralized ledger system storing data in a plurality of data blockchains, each of the plurality of data blockchains comprising a plurality of data blocks storing transaction data;

storing the trust timestamp and the associated signature from the trust time server in a centralized trust timestamp blockchain, the centralized trust timestamp blockchain comprising a plurality of timestamp blocks storing trust timestamp information of the trust time server for the centralized ledger system; and

maintaining the centralized trust timestamp blockchain in the centralized ledger system by sequentially storing trust timestamps and associated signatures received from the trust time server in respective timestamp blocks according to a chronological order of receiving the trust timestamps and associated signatures from the trust time server.