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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 March 2017 (23.03.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/048630 A1

(51) International Patent Classification:
G06F 21/64 (2013.01) *H04L 9/08* (2006.01)
G06Q 30/02 (2012.01)

(21) International Application Number:
PCT/US2016/051301

(22) International Filing Date:
12 September 2016 (12.09.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
14/852,955 14 September 2015 (14.09.2015) US

(71) Applicant: **MEDICI, INC.** [US/US]; 6350 South 3000 East, Salt Lake City, Utah 84121 (US).

(72) Inventors: **BLACK, Tron**; 6350 South 3000 East, Salt Lake City, Utah 84121 (US). **WILKINS, Alec**; 6350 South 3000 East, Salt Lake City, Utah 84121 (US). **CHRISTENSEN, Robert**; 6350 South 3000 East, Salt Lake City, Utah 84121 (US).

(74) Agents: **RASMUSSEN, Laurie M.** et al.; Perkins Coie LLP, P.O. Box 1247, Seattle, Washington 98111-1247 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

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

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

(54) Title: DATA VERIFICATION METHODS AND SYSTEMS USING A HASH TREE, SUCH AS A TIME-CENTRIC MERKLE HASH TREE

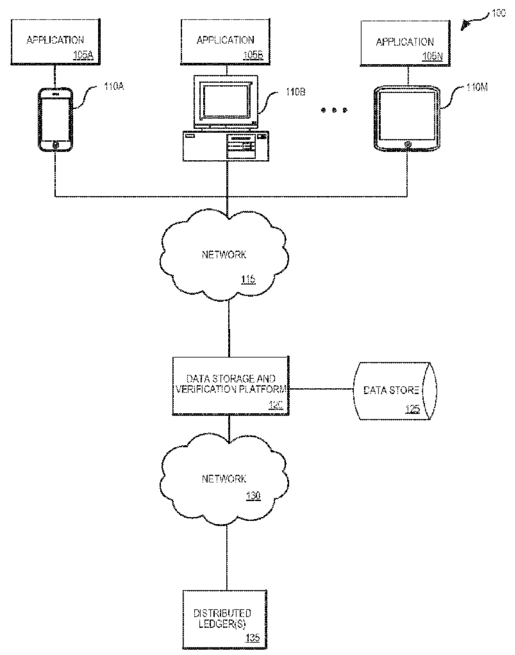


FIG. 1

(57) Abstract: Systems and methods described herein generally relate to storing and verifying data. In some embodiments, reference levels are generated according to time intervals, where the first reference level comprises a predetermined number of the time intervals, and where each of the time intervals of the remaining reference levels is comprised of a predetermined number of the time intervals of a previous reference level. Hashes of data can be created at the first reference level by performing a hashing function on the data in a time-sequenced manner. First reference level time interval hashes may be generated by performing the hashing function on the hashes of the data at each of the time intervals of the first reference level. Hashes for remaining reference level time intervals can be generated by performing the hashing function on the hashes of each of the time intervals of the previous reference level.

WO 2017/048630 A1

DATA VERIFICATION METHODS AND SYSTEMS USING A HASH TREE, SUCH AS A TIME-CENTRIC MERKLE HASH TREE

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to and benefit from United States Patent Application No. 14/852,955, filed on September 14, 2015, entitled "DATA VERIFICATION METHODS AND SYSTEMS USING A HASH TREE, SUCH AS A TIME-CENTRIC MERKLE HASH TREE," the entire content of which is hereby incorporated by reference for all purposes in its entirety.

TECHNICAL FIELD

[0002] Various embodiments of the present disclosure generally relate to storing and verifying data. More specifically, various embodiments of the present disclosure relate to systems and methods for storing and verifying data using hashing techniques.

BACKGROUND

[0003] A hash function is a function that can be used to map digital data of arbitrary size to digital data of fixed size. Hash functions can be used for many purposes, such as to accelerate table or database look-up by detecting duplicated records in a large file. Hash functions are also used in blockchains. Blockchains are verifiable permanent ledgers constructed one block at a time with a proof-of-work seal (hash) affixed to each block that validates that block. In any blockchain, the hash of the previous block is included in the current block, and therefore by recursion the current hash also validates all previous blocks back to the original genesis block. Inserting a hash into a blockchain permanently records that hash and acts as a notary verifying the timestamped proof of existence of the hashed data at the moment in time that the block is added to the chain. Future blocks add a layer of protection from a chain re-organization and therefore add certainty that no changes can be made to blocks earlier in the chain.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] Embodiments of the present disclosure will be described and explained through the use of the accompanying drawings.

[0005] Fig. 1 illustrates an example of a network-based operating environment in accordance with various embodiments of the disclosure.

[0006] Fig. 2 illustrates a set of components in a data storage and verification platform according to one or more embodiments of the present disclosure.

[0007] Fig. 3 is a diagram illustrating an architecture of a data storage and verification platform according to one or more embodiments of the present disclosure.

[0008] Fig. 4 illustrates a process of storing and verifying data using a non-sparse TOME according to one or more embodiments of the present disclosure.

[0009] Fig. 5 illustrates a process of storing and verifying data using a sparse TOME according to one or more embodiments of the present disclosure.

[0010] Fig. 6 illustrates a process of storing and verifying data using a clockchain TOME according to one or more embodiments of the present disclosure.

[0011] Fig. 7 illustrates an example of a computer system with which some embodiments of the present disclosure may be utilized.

DETAILED DESCRIPTION

[0012] Various embodiments of the present disclosure generally relate to storing and verifying data. More specifically, various embodiments of the present disclosure relate to systems and methods for storing and verifying data using hashing techniques.

[0013] The Data Storage and Verification Platform describes a method and system in which a record of data, and particularly rapidly changing, time-sensitive data, can be generated both quickly and efficiently using the concept of a Merkle Tree where the tree branches are time-centric groupings of data.

[0014] Traditional methods of data validation are inefficient, particularly where large amounts of rapidly changing data are concerned (e.g., trading data, telemetry). For example, one solution for verifying data includes storing the entire corpus of a file or text and comparing it with the original data to confirm validity. While this method is manageable for small amounts of data, this solution is impractical for a comparison of any significant amount of data.

[0015] Another current solution is to store the data in a blockchain. But recording rapidly changing data (e.g., each trade on an exchange) in a blockchain is impractical for at least two reasons. First, the amount of data that must be synchronized and stored currently exceeds most communication channels of, most storage systems of, and the "bandwidth" of current blockchains. Put simply, rapidly changing data cannot be synchronized quickly across a widely distributed, decentralized system. Second, it is impractical to record rapidly changing data on a blockchain because the timing of a decentralized blockchain is not deterministic. That is, pathways for the data are dependent on the peer-to-peer connections which may lead to rapidly changing data being recorded in a different order than it originally occurred.

[0016] Methods and systems described herein provide a way for data, including rapidly changing, time-sensitive data, to be recorded for verification in the future. Some embodiments described herein describe a Data Storage and Verification Platform that can generate a Time Ordered Merkle Tree Epoch (TOME), where "time" may refer to an indefinite continued progress of existence and events in the past, present, and future regarded as a whole; where "ordered" may refer to being arranged in a methodical or appropriate way; where "Merkle Tree" refers to the Merkle Tree invented by Ralph Merkle, in which each non-leaf node is labeled with the hash of the labels of its children nodes; and where "epoch" refers to the beginning of a distinctive period in the history of someone or something." A TOME may use a cryptographic hash function such as SHA256 for the hashing.

[0017] In some embodiments, the leaves of the TOME may be specific records and the tree may be defined by the number of data records per the first reference level, the number of segments or time intervals in the first reference level that compose the second reference level, the number segments or time intervals in the second reference level that compose the third reference level, and so on. In some embodiments, the reference levels are defined by time, and in such embodiments, the tree can be defined by the number of data records per second, the number of seconds per minute, the number of minutes per hour, the number of hours per day, etc.

[0018] In an example and as further described herein, the Data Storage and Verification Platform can receive data records and provide a timestamp if necessary. Each data record may be hashed with a corresponding timestamp generated by the Data Storage and Verification Platform on reception of the data record. The timestamp can represent varying levels of granularity (e.g., picosecond, second, etc.). In one illustrative example the granularity is represented in seconds, and thus, the hashes for each data record are combined in a temporal/sequential order and a hash of the hashes is generated each second; the combined hash may be referred to as a one-second TOME. Thus, a hash is created for each second. After sixty one-second hashes have been created, all of the one-second hashes are then combined (ascending by time) and hashed to create a one-minute hash, which may be referred to as the one-minute TOME. After sixty of the one-minute hashes have been created, all of the sixty one-minute hashes are ordered (ascending by time) and hashed to create the one-hour TOME. After twenty-four of the one-hour hashes have been created, the twenty-four one-hour hashes are ordered (ascending by time) and hashed to generate the one-day TOME. The day hashes can then be ordered (ascending by time) and hashed to make a one-month TOME (i.e., a hash of thirty or thirty-one daily hashes), a quarterly TOME (i.e., a hash of ninety daily hashes), and/or a yearly TOME (i.e., a hash of 365 daily hashes).

[0019] Combining the hashes before a hashing function is performed on the combined hashes can be done using various methods. For example, the hashes can be combined by concatenating the hashes together, by summing the hashes numerically, or by concatenating the binary bytes together. Other methods might also XOR the hashes together, concatenate the hashes with a delimited between them, or add an index number to each hash and then combine the hashes. The method of combining the hashes must be known in order for an independent party to replicate the process and independently arrive at the same result.

[0020] There are several types of TOMEs, including a non-sparse TOME, a sparse TOME, and a clockchain TOME. In a non-sparse TOME, hashes for each segment (or unit of time, e.g., one second) are generated regardless of whether data was received during a segment of the first reference level (e.g., a second). Thus, in a non-sparse TOME, there will always be a hash for each segment (i.e., sixty one-second hashes will be created and used to create the one-minute hash even if data was received in only three or four of the sixty one-second hashes).

[0021] In a sparse TOME, a hash is created only if data was received during the segment of the first reference level (i.e., if data is received in three seconds out of sixty in a particular minute, only three hashes will be hashed to form the one-minute hash).

[0022] A clockchain TOME uses a genesis hash. The genesis hash is the first hash of the epoch and can be a hash of a document that provides a description of the data being hashed or any other type of information. The genesis hash is hashed with the data records in the first segment (e.g., second). In this embodiment, the hash for the first segment (e.g., first second) is hashed with the data records in the next segment (e.g., second second) to create the second segment hash, the second segment hash is hashed with the hash of the data records received during the third segment (e.g., third second), and so on.

[0023] The hashes can be recorded in a blockchain at any point, allowing the data to be verified in the future. That is, by reconstructing the TOME with the data, the same hash will be created if the data is undamaged and unaltered. However, if any of the data is changed or if a timestamp in the data has changed, the hash will be different, indicating a difference in the data.

[0024] The hashes created using the Data Storage and Verification Platform can be publicly attested to using cryptographic techniques such as public-key cryptography and bidirectional encryption. Public-key cryptography requires a key pair, where the two keys are mathematically linked. One key is a public key that is freely shared among nodes in a peer-to-peer network. The other key is a private key that is not shared with the public. The public key is used to encrypt plaintext and to verify a digital signature. The private key is used to decrypt cipher text and to digitally sign messages. The message may be digitally signed by the sender's private key to authenticate the sender's identity. Then, the sender's digitally signed transaction message may be decrypted using the sender's public key to verify that the sender originated the message.

[0025] Benefits of the Data Storage and Verification Platform include transparency and immutability, particularly in time-sensitive data, because the Data Storage and Verification Platform can determine whether data has been altered in the slightest when the data is run through the same algorithm and compared to a stored hash. The Data Storage and Verification Platform provides non-repudiation when hashes of the data are recorded to the blockchain. Once a hash has been recorded in the blockchain, the hashed data cannot be tampered with without invalidating the hash. The Data Storage and Verification Platform provides external attestation by allowing for a digital signature attesting to the validity of the hashed data (e.g., public key cryptography).

[0026] Data records may be verified in any time interval (e.g., second, day, week, month, year, decade). The Data Storage and Verification Platform can validate data in smaller time intervals, avoiding the need to verify an entire data set which may span decades. The Data Storage and Verification Platform can validate any type of data or any amount of data. Additionally, storage required for the data record is insignificant. While it is impractical to record each trade on a distributed ledger, hashes at certain points in time (e.g., hourly, daily, weekly, monthly) can be recorded to distributed ledgers (e.g., blockchain for Bitcoin) that are maintained by network nodes.

[0027] Time, and specific time increments, are used as examples in the disclosure. However, embodiments of the disclosure can be used with any data and hashed in any time interval.

[0028] The techniques introduced here can be embodied as special-purpose hardware (e.g., circuitry), as programmable circuitry appropriately programmed with software and/or firmware, or as a combination of special-purpose and programmable circuitry. Hence, embodiments may include a machine-readable medium having stored thereon instructions that may be used to program a computer (or other electronic devices) to perform a process. The machine-readable medium may include, for example, floppy diskettes, optical disks, compact disc–read-only memories (CD-ROMs), magneto-optical disks, read-only memories (ROMs), random access memories (RAMs), erasable programmable read-only memories (EPROMs), electrically erasable programmable read-only memories (EEPROMs), magnetic or optical cards, flash memory, or other type of media/machine-readable medium suitable for storing electronic instructions.

[0029] Fig. 1 illustrates an example of a network-based operating environment 100 in which some embodiments of the present disclosure may be used. As illustrated in Fig. 1, operating environment 100 includes applications 105A-105N running on one or more computing devices 110A-110M (such as a mobile device, a mobile phone, a tablet computer, a mobile media device, a mobile gaming device, a vehicle-based computer, a dedicated terminal, a public terminal, a desktop or laptop computer, a smartwatch or other wearable technology, a kiosk, etc.). In some embodiments, applications 105A-105N for carrying out operations such as generating documents or orders may be stored on the computing devices or may be stored remotely. These computing devices can include mechanisms for receiving and sending traffic by connecting through network 115 to the Data Storage and Verification Platform 120.

[0030] Computing devices 110A-110M are configured to communicate via network 115 with Data Storage and Verification Platform 120. In some embodiments, computing devices 110A-110M can retrieve information from or submit information to Data Storage and Verification Platform 120 or data store 125 and run one or more applications with customized content retrieved by Data Storage and Verification Platform 120. For example, computing devices 110A-110M each can execute a browser application or a customized client to enable interaction between the computing devices 110A-110M and Data Storage and Verification Platform 120.

[0031] Data Storage and Verification Platform 120 can run on one or more servers and can be used to create data records and verify data using hashing techniques, to record hashes to a distributed ledger, to record digital signatures, and to compare hashes, among other activities. Data Storage and Verification Platform 120 may be communicably coupled with data store 125 and computing devices

110A-110M and may communicate, access or receive data (e.g., documents, trade data) from computing devices 110A-110M and data store 125.

[0032] Data Storage and Verification Platform 120 may be customized or calibrated by individual companies or service providers based on user needs and/or business objectives. For example, the interval at which the data is hashed may be based on various time intervals in which the data is received (e.g., trade data arrives faster than a second whereas monthly reports generally occur monthly) and thus Data Storage and Verification Platform 120 may be calibrated differently for different uses and different users.

[0033] Data Storage and Verification Platform 120 provides, among other things, a method of creating a record of data such that the data can be easily validated at any time in the future. The Data Storage and Verification Platform 120 may create TOMEs in which the leaves of the tree are specific records and the tree is defined by the number of data records per first segmented time period, the number of segments in the first time period that compose the second segmented time period, the number of segments in the second segmented time period that compose the third segmented time period, and so on. For example, the tree may be defined by the number of data records per second, the number of seconds per minute, the number of minutes per hour, the number of hours per day, etc. Data Storage and Verification Platform 120 may record hashes at any point in the tree and then compare that record to hashes of data to verify that the data has not changed.

[0034] Data store 125 can be used to manage storage and access to data such as trade data, documents, user information, and other information. Data store 125 may be a data repository of a set of integrated objects that are modeled using classes defined in database schemas. Data store 125 may further include flat files that can store data. Data Storage and Verification Platform 120 and/or other servers may collect and/or access data from the data store 125.

[0035] Data Storage and Verification Platform 120 is communicably coupled with one or more distributed ledger(s) 135 through network 130.

[0036] Network 115 and network 130 can be the same network or can be separate networks and can be any combination of local area and/or wide area networks, using wired and/or wireless communication systems. Either network 115 or network 130 could be or could use any one or more protocols/technologies: Ethernet, IEEE 802.11 or Wi-Fi, worldwide interoperability for microwave access (WiMAX), cellular telecommunication (e.g., 3G, 4G, 5G), CDMA, cable, digital subscriber line (DSL), etc. Similarly, the networking protocols used on network 115 and network 130 may include multiprotocol label switching (MPLS), transmission control protocol/Internet protocol (TCP/IP), User Datagram Protocol (UDP), hypertext transport protocol (HTTP), simple mail transfer protocol (SMTP) and file transfer protocol (FTP). Data exchanged over network 115 and network 130 may be represented using technologies, languages and/or formats including hypertext markup language (HTML) or extensible markup language (XML). In addition, all or some links can be encrypted using conventional encryption technologies such as secure sockets layer (SSL), transport layer security (TLS), and Internet Protocol security (Ipsec).

[0037] Distributed ledger(s) 135 records hashes either automatically (e.g., at the end of a time period) or as requested on a distributed ledger. For example, Bitcoin uses a distributed public ledger called the blockchain. When distributed ledger(s) 135 receives a hash signed with the proper key from Data Storage and Verification Platform 120 and the hash is verified by network nodes, distributed ledger(s) 135 records the hash to the distributed ledger.

[0038] Fig. 2 illustrates a set of components within Data Storage and Verification Platform 120 according to one or more embodiments of the present disclosure. According to the embodiments shown in Fig. 2, Data Storage and Verification Platform 120 can include memory 205, one or more processor(s) 210, data receiving module 215, time stamping module 220, hashing engine 225, recording module 230, digital signature module 235, data validation module 240, comparing module 245, and Graphical User Interface (GUI) generation module 250. Other embodiments may include some, all, or none of these modules and components along with other modules, applications, and/or components. Still yet, some embodiments may incorporate two or more of these modules and components into a single module and/or may associate a portion of the functionality of one or more of these modules with a different module. For example, in one embodiment, data validation module 240 and comparing module 245 can be combined into a single component.

[0039] Memory 205 can be any device, mechanism, or populated data structure used for storing information. In accordance with some embodiments of the present disclosure, memory 205 can be or include, for example, any type of volatile memory, nonvolatile memory, and dynamic memory. For example, memory 205 can be random access memory, memory storage devices, optical memory devices, magnetic media, floppy disks, magnetic tapes, hard drives, erasable programmable read-only memories (EPROMs), electrically erasable programmable read-only memories (EEPROMs), compact discs, DVDs, and/or the like. In accordance with some embodiments, memory 205 may include one or more disk drives or flash drives, one or more tables, one or more files, one or more local cache memories or processor cache memories, one or more relational databases or flat databases, and/or the like. In addition, those of ordinary skill in the art will appreciate many additional devices and techniques for storing information that can be used as memory 205.

[0040] Memory 205 may be used to store instructions for running one or more applications or modules on processor(s) 210. For example, memory 205 could be used in one or more embodiments to house all or some of the instructions needed to execute the functionality of data receiving module 215, time stamping module 220, hashing engine 225, recording module 230, digital signature module 235, data validation module 240, comparing module 245, and GUI generation module 250.

[0041] Data receiving module 215 can receive data (e.g., data items, data records) into the Data Storage and Verification Platform 120 to create a record of the data using hashing techniques. The data can be any type of data (e.g., document, video, picture, email message), and the data can be received at any interval (e.g., every second, every hour, three times a year, randomly). In some embodiments, the data received by data receiving module 215 can be kept secret but still validated. For example, the data that is sent could be a hash of secret data providing a way for non-repudiation without transparency. In some embodiments, the data received by data receiving module 215 can be the result of an API call (provided that the result of the API call stays consistent for validation of third-party data).

[0042] Data receiving module 215 can also receive verification data for the purpose of verifying that the verification data is the same as the data that was initially received. That is, a record of verification data may be made using the same hashing techniques to determine whether it is the same as the initial data. Such verification may be helpful in verifying data transferred from peer to peer networks, to detect fake data, or to comply with an audit. Data or data items received via data receiving module 215 may be communicated to time stamping module 220 for time stamping, if needed, or directly to hashing engine 225 for hashing if time stamping is unnecessary.

[0043] Time stamping module 220 can determine whether a timestamp is needed and then time stamp the data as needed. Some data may already include a timestamp, which may be helpful for time-sensitive data. In some embodiments, the data is timestamped regardless of whether the data already includes a timestamp. The data can be timestamped at any granularity (e.g., picoseconds, milliseconds, microseconds, nanoseconds, seconds, minutes, hours). The timestamp can be the beginning of the epoch or a real timestamp of the data record. The epoch value passed is the number of seconds encompassed in the epoch. The time can be specified in any manner specific enough to denote a moment in time (worldwide) (e.g., Julian, data string with a time zone such as GMT). The data, with a timestamp, is sent to hashing engine 225.

[0044] Hashing engine 225 receives data and hashes the data with its timestamp at its time-stamped record level. Then the hashes for each record are combined in time order. Sometimes, the time resolution is not granular enough to put the records in a deterministic order (i.e., two data records are received at the same time or are recorded with the same time stamp). During such circumstances, the hash of the data records can be used for a secondary sort order because during verification, the hash of the verification data will result in the same hash as the original data (if the data is the same), which can be sorted. Thus, in some embodiments, the data is sorted by time first, then, if there is a tie, the data is sorted by the hashes of the data. Sorting in this manner allows a third party or a different compatible system a method of ordering the data in a consistent manner.

[0045] Each reference level may have a number of sequenced segments, each of which are hashed to create a single hash for the reference level, which is then used in generating the next reference level. As an example, assume that seconds are the first reference level, minutes are the second reference level, and hours are the third reference level. The first reference level includes 60 sequenced segments (i.e., 60 seconds), the hashes of which will be used as a segment of the second reference level. As the data is received, the data is hashed, and all the data received during a particular segment (i.e., second) is hashed. Then the hashed data for each segment (i.e., 60 hashes) is hashed together in an ascending order to generate one segment on the second reference level (i.e., 60 one-second hashes are hashed to generate a one-minute hash). The second reference level includes 60 segments (i.e., 60 minutes). Thus, 60 sequenced segments from the second reference level are hashed together in an ascending order to generate one segment on the third reference level (i.e., 60 one-minute hashes are hashed to create a one-hour hash). The third reference level includes 24 segments which are hashed together in an ascending order to generate one segment on the fourth level (i.e., 24 one-hour hashes are hashed to create a one-day hash). This process can continue such that one hash is representative of weeks, months, quarters, years, decades of data, or any other time period.

[0046] Each hashed segment of the record data at each reference level may be labeled as a hashed segment for the reference level, or a TOME. Thus, if the first reference level is a second, the hash for each second is a hashed segment for the second reference level. If the second reference level is a minute, the hash of sixty of the hashed second segments is one hashed segment for the minute reference level. If the third reference level is an hour, then the hash of sixty of the hashed minute segments is one hashed segment for the hour reference level.

[0047] When data is received, the data is timestamped and hashed with the timestamp. Then all of the data that is received within the first second is hashed together, in a time-sequenced manner. The same process occurs for the second second, the third second, and so on until the end of the segments in that reference level.

[0048] Hashing engine 225 can use slightly varied hashing algorithms. For example, in a "sparse TOME," when data is not received during any period of time, no hash is created or stored for that time period. Thus, it is possible that there is no hash for a particular second, minute, hour, or week. In a sparse TOME, if data at a time period in which no data was received is asked to be validated, it will return *null*, 0, or *nothing* for the hash. For example, in a sparse TOME that has only one data point in a year, there would be data for that second, minute, hour, day, week, month, and year, but nothing else (e.g., sha256(sha256(sha256(sha256(sha256(sha256(one_data_point))))))).

[0049] Hashing engine 225 can also use a "non-sparse TOME" algorithm. In a non-sparse TOME algorithm, if there is no data at any point in time, hashing engine 225 still creates a hash. Assuming seconds is the lowest reference level, if no data is received during a second, a hash is created for the second (e.g., a hash of an empty string). For example, the static SHA256 hash of the empty string can be used. In a non-sparse TOME, there will always be 60 one-second hashes to generate the one-minute hash.

[0050] There are advantages to using the sparse TOME over non-sparse TOME and vice versa. For example, the non-sparse TOME provides a data point confirming that no data was received during the time period, whereas there is no specific confirmation except a lack of data when using the sparse TOME. On the other hand, because the non-sparse TOME provides a hash regardless of whether data is received, the hash file for a non-sparse TOME will be larger than the hash file of a sparse TOME when data is not received during every segment of the first reference level.

[0051] Generally, data storage for the hashes is not substantial. For example, storage for an entire year of non-sparse TOME data may be roughly two gigabytes of data, assuming that the reference levels include seconds, minutes, hours, weeks, months, and years; that the hashes are SHA256; that the binary hashes are stored as 32-byte hashes; and that the concatenated hashes are stored as 64-hex characters. Storage numbers may increase or decrease depending on how much data is received and hashed for a given second (e.g., the hex string that is hashed for a particular second may be larger or smaller, depending on the number of data items received during the second). In a non-sparse TOME, the minute hash will always be a 3840 (i.e., 64*60) character string of hex values where a weekly hash is a 448 (i.e., 64*7) character string of hex values of the daily hashes.

[0052] Hashing engine 225 can also use a "clockchain TOME." The clockchain TOME is a non-sparse TOME in that there are hashes for each segment of the lowest reference (e.g., each second) regardless of whether data is received for the segment. The clockchain TOME is unique from the non-sparse TOME in that it takes the hash from a previous segment of the first reference level, uses the hash from the previous segment as the first hash for the current segment, and hashes the hash from the previous segment with the other data hashes received for the current segment, in ascending order. The hash of the current segment is then used as the first hash for the following segment.

[0053] The very first hash of the very first segment of the TOME may be called a genesis hash as it is a hash of the genesis meta-data. The genesis meta-data may define terms relating to the TOME data and/provide the reasoning for the TOME's existence (e.g., document specifying legal definition of an asset being traded and the original quantity, customer number, generation of broker-dealer), whereas the TOME data is the hashes of the actual data (e.g., trade data for the asset). Thus, the genesis hash and all the data hashes received during the first second are hashed to form the first second hash, the first second hash is hashed with all the data hashes received during the second second to form the second data hash, the second data hash is hashed with all the data hashes received during the third second, and so on. The clockchain essentially has a birthday and tracks every data point for every second, beginning with the birthday of the clockchain tree. Additionally, during validation, the clockchain TOME will validate (and is required to validate) every hash going backwards to the beginning of the tree. Thus, the clockchain could validate every trade from the beginning of a stock's issuance.

[0054] Hashing engine 225 may create the hashes using SHA256, and may represent the hashes using 64 hex characters. Storage may be optimized by storing the 32-byte binary hashes, but the concatenated hashes to be hashed can be the 64 hex characters.

[0055] Recording module 230 records one or more of the hashes (TOMEs) into a medium such as a blockchain. Any blockchain can be used to record one or more TOMEs. Recording module 230 can record the TOMEs at any time interval that a commit occurs (e.g., every minute, every hour by combining the minute hashes). A commit is made specifying the end of an epoch boundary (e.g., end of a second, end of a minute) and locks the data so no more data may be added once committed. Then recording module 230 combines all the hashes up to the epoch boundary. If the commit occurs on a timestamp that is not a boundary (e.g., 1.5 seconds), then the last boundary going backwards may be used (e.g., 1 second).

[0056] The time interval for recording the TOMEs may depend on the speed of the blockchain. For example, some blockchains may record hour hashes into a ten-minute blockchain, such as Bitcoin, or minute hashes every minute in a 30-second blockchain such as Inifinitecoin. In some embodiments, the TOMEs are automatically recorded in the blockchain at certain intervals. For example, the hourly or daily times that encompass the entire data for that epoch may be recorded. Thus, recording module 230 records the hash to transparently and permanently record proof of existence of the data for any chosen block of time.

[0057] In some embodiments, after the data is recorded through a certain epoch (e.g., a day), no changes may be made to any of the TOMEs included in that day (e.g., daily). Other mediums besides blockchains can be used to record the hashes such as a newspaper, website blog, twitter, etc.

[0058] Digital signature module 235 can allow multiple entities to sign (i.e., attest to) the data at any time level. Attestations could occur daily, monthly, weekly, and/or yearly. To attest to the data, public key cryptography can be used such that an ATS or broker-dealer can digitally sign a hash, publicly stating that the hash of the data is authentic and represents the actual data. Additionally, digital signatures could be affixed automatically by the devices that independently audit the data and verify the hashes. In some embodiments, each data entry may be digitally signed. In an example, a hash of the data is played through two systems and thereafter the hashes for the day are compared. If the hashes are the same, then a digital signature can be applied to the hash validating the data for the day. In some embodiments, hashes are digitally signed before being recorded on the blockchain.

The signature of a hash represents that all time-ordered data that occurred during that epoch is valid. Thus, a signature of a hash for a week of data attests that the data for the entire week is valid.

[0059] Data validation module 240 validates verification data (i.e., determines whether the data has changed since the original data was hashed) by receiving a TOME at a point in time and running the potentially suspect verification data through the same hashing algorithm. The verification data would include the data from the same time range as the initial data. Assuming that the same hashing algorithm or function with the same specifications (e.g., how time records are specified, the precisions of real numbers, data format (e.g., JSON, XML, Binary)) is used, the same hashes should be replicated for each data record, resulting in the same TOMEs. Because the timestamp is recorded in the record, the Merkle Tree hashes for the second, minute, day, hour, etc. can also be replicated. Data validation module 240 can validate any type of data. Thus, data validation module 240 can verify and guarantee that data has not been added, removed, or tampered with for any epoch (i.e., time range). In some embodiments, data validation module 240 performs a verification of data upon a triggering event. Triggering events may include an audit event required by a regulator or other entity requesting a data integrity check. Data validation module 240 may be used to compare one system with another system that was running in parallel.

[0060] The amount of data required for the validation depends on the type of algorithm used to create the hashes and the timeframe of the data for validation. For example, to validate data hashed using the clockchain algorithm, data validation module 240 must validate the entire timeframe requested for validation, beginning at the genesis meta-data. Thus, the user must provide the genesis meta-data, as well as all the data leading up until the point of time that the user is interested in validating. For every validation that is performed, by design, data validation module 240 validates every hash going backwards to the beginning of the tree when the clockchain algorithm began. In contrast, when using a non-sparse TOME or a sparse TOME, the user can specify a subset of data to validate. Since there is no particular genesis meta-data or birthday for the non-sparse TOME or the sparse TOME, it is not necessary to start at a particular point in time. For example, if the user wants to validate a particular hour of data, then the data for that entire hour is all that is necessary. Data validation module 240 takes the data for the hour and recreates the hashes for the hour. Similarly, if the user wants to validate a year of data, only the data for that entire year is necessary.

[0061] Comparing module 245 compares the original hash with the hash of the verification data created by data validation module 240. If the hashes are different, something in the data was changed. Any change in the data, including a change of a timestamp, a change to the data, or a re-ordering of data, will produce a different hash. For example, if the data is a movie and one pixel of the movie is changed, the hash will be different. In another example, if the data is trade data and a timestamp of one trade out of millions of trades has been altered, the hash will be different.

[0062] If the hash is different, then something in the data changed. To determine precisely where the data was altered, a smaller amount of time may be examined. For example, in a non-sparse TOME or a sparse TOME, if a month of data was validated and the validation failed, then each of the weeks of that month of data could be validated to determine which week failed. Once a failed week is determined, each day in the failed week could be validated to identify which day failed. Once the failed day is identified, each hour could be validated to determine which hour failed. This process can be iteratively continued until the exact data hash that failed is identified. If the algorithm is a clockchain, then identifying the exact data that has changed will be a slightly different process because all the data from the genesis up to the point of validation has to be included. Assuming that a month of data failed to validate, all the data from the genesis to the end of the month, and the data from the genesis to the middle of the month, could be validated to determine whether the data changed during the first half of the month or the second half of the month. Once it is determined which half of the month contained the altered data, then the next iteration can validate up to the first week of the month (if the first two weeks contained the bad data) or the third week of the month (if the last two weeks contained the bad data). An iterative process as described can be performed until the exact data record is identified.

[0063] By validating the order of events, no re-ordering of events can occur without invalidating the verification data. The comparison data may be provided to auditors to prove that the data has not changed. Thus, the high-speed and high-volume nature of modern data flows can be matched with the traditional auditing intervals of monthly, quarterly, or yearly, as required by regulations.

[0064] GUI generation module 250 can generate one or more GUI screens that allow interaction with a user. In at least one embodiment, GUI generation module 250 generates a graphical user interface receiving and/or conveying information to the user. For example, GUI generation module 250 may display a user interface in which the user can request timeframes of data to be validated and specify the type of algorithm used. GUI generation module 250 can also display the validation results.

[0065] Fig. 3 is a diagram 300 illustrating interaction of components used in a data storage and verification platform. As shown in Fig. 3, Data Feed 302 receives high-speed data of ordered discrete data items. TOME 304 generates TOMEs of the high-speed data by hashing the data items in a sequenced manner using reference levels. Recorder 306 records hashes of the data items at any point in time on a blockchain such that the a record of the hash is memorialized. Thus, data can be verified as being unchanged by later by reconstructing the recorded hash with the data and verifying that the recorded hash and the reconstructed hash are the same. Attestation unit 308 receives digital signatures verifying that data and/or hashes are accurate (i.e., the data is uncompromised). The attestations may be used later as further proof that the initial data was the correct data.

[0066] Fig. 4 illustrates a process of storing and verifying data using a non-sparse TOME. As shown, data items 402, 404, 406, 408, and 410 are received, and each data item is timestamped and hashed. As noted above, in a non-sparse TOME, even if no data is received during a second, a hash will be created specifying that there was no data during the second. Next, the hashes of the data items for each second are hashed together, sequentially. For example, data items 402 and 404 are received during the first second; thus, the hashes of data items 402 and 404 are hashed together to create the hash of the first second of the first set of second hashes, element 412. Fig. 4 shows only two data items occurring during the first second; however, fewer or more data items could be received, and each will be individually hashed and subsequently hashed with the other hashes of the data items. This process is repeated for the following 59 seconds such that there are 60 hashes representing all the data items received during each second.

[0067] Thereafter, the 60 hashes (one for each second) are hashed together to create the first minute hash, element 416. When a leap-second is applicable, 61 second hashes may be hashed together to create a minute hash. This process is repeated for 60 minutes such that at the end of 60 minutes there are 60 hashes representing all the data received during each of the minutes. As shown, data items 406, 408, and 410 are hashed during a second set of seconds (e.g., seconds 61-120). Then the hashes of the data items 406, 408, and 410 are hashed in element 414 to generate the first of the second set of second hashes. Once hashes for all 60 seconds (e.g., seconds 61-120) are created, a second minute hash, element 418, is created by hashing all 60 of the second hashes.

[0068] Next, elements 416 and 418, each hashes of a minute of data, are hashed together with 58 other hashes of a minute of data to generate a first hour hash, element 420 (i.e., 60 minutes of hashed data). This process is repeated twenty-three times such that there are a total of 24 hour hashes. Each of the 24 hour hashes are hashed together to create a first day hash, element 422. Day hashes can be created and hashed together in any desirable increment. For example, 365 day hashes can be hashed to generate a hash for an entire year of data, element 424. While 365 day hashes are shown in the element 424, it is understood that the number of days could be 365 or 366, depending on the number of days in the particular year. Or, 30 day hashes can be hashed to generate a hash for a month, element 426. While 30 day hashes is shown in element 426, the number days could be 28-31, depending on the number of days in a particular month. In some embodiments, instead of hashing 365 day hashes, twelve month hashes can be hashed together to create a year hash. Additionally, 90 day hashes (or 91 or 92 day hashes, as applicable for the quarter) can be combined to create a hash for a quarter, element 428.

[0069] Fig. 5 illustrates a process of storing and verifying data using a sparse TOME. The sparse TOME works in the same way as the non-sparse TOME except that when data is not received during a time period, a hash is not created. Thus, in a sparse TOME, there may be many missing hashes, depending on how fast the data is being received. As shown, data is received during the first second (data items 502 and 504) and the twenty-fifth second (data items 506, 508, and 510). The data hashes for the first second, element 512, and for the twenty-fifth second, element 514, are each hashed, and the hash of the first second is hashed with the twenty-fifth second to generate the first minute hash, element 516. Since no data items were received during seconds 2-24 or 26-60, no additional hashes are created. In contrast, in a non-sparse TOME, regardless of whether data was received during a second, the system generates a hash such that there will always be 60 hashes to be hashed to generate the minute hash.

[0070] The hour hash is then created, element 518, by hashing the minute hashes. In this example, there are minute hashes available for the first minute (M1) and the fourth minute (M4), meaning that at least some data was received during the first minute and the fourth minute. Next, a day hash is created by hashing the hash of each hour in which data was received. Here, the day hash, element 520, includes a hash of the first hour, a hash of the fifth hour, and a hash of the twenty-third hour. A hash of the data for the year can be generated by hashing the hashes of the available days (e.g., the first day and the second day), element 522.

[0071] Fig. 6 illustrates a process of storing and verifying data using a clockchain TOME. The clockchain TOME operates identically to the non-sparse TOME with the exception of two features. First, every clockchain TOME begins with a genesis hash, element 602. The genesis hash could be any type of data, preferably data that explains what data is included in the TOME (e.g., stock identifier, customer number). The genesis hash and the hashes of data items 604 and 606 are hashed together to generate the first second hash, element 614.

[0072] The second difference from the non-sparse TOME is that the second second hash begins with the first second hash, element 608. Thus, the second second hash, element 616, is a hash of the first second hash, element 608, and hashes of data items 610 and 612. This process is repeated for each second. For example, the second second hash would be the first hash of the third second hash. Once the first sixty seconds of hashes are complete, the hashes for each of the sixty seconds (or 61 seconds when there is a leap second) are hashed to create the first minute hash, element 618. Then the data received in the following sixty seconds is hashed in the same manner to create the second minute hash. This process is repeated for each of the following minutes for a total of sixty minutes. The sixty minute hashes are hashed to create an hour hash, element 620. Elements 622, 624, 626, and 628 are created in the same manner described for elements 422, 424, 426, and 428, respectively.

[0073] Various embodiments of the present disclosure are described below.

1. A computerized method comprising:

    receiving data items during first reference level segments of a first reference level;

    performing a hashing function on each of the data items with a corresponding timestamp;

    generating a first reference level segment hash for each of the first reference level segments by performing a second hashing function on the hashes of the data items in each of the first reference level segments according to their respective timestamps; and

    generating a second reference level segment hash for each of multiple second reference level segments, wherein the each of the multiple second reference level segments is comprised of a predetermined number of the first reference level segments, wherein the each of the multiple second reference level segment hashes are generated by performing a third hashing function on the predetermined number of the first reference level segment hashes.

2. The computerized method of claim 1, further comprising recording at least one of the second reference level segment hashes to a distributed ledger.

3. The computerized method of claim 1 or 2, wherein the first reference level segments and the second reference level segments are periods of time.

4. The computerized method of claim 3, wherein the period of time of the first reference level segment is a second, and wherein the period of time of the second reference level segment is a minute.

5. The computerized method of claim 4, wherein the predetermined number of the first reference level segments is sixty or sixty-one.

6. The computerized method of claim 1, 2, 3, or 4, further comprising generating reference level segment hashes for additional reference levels, wherein each of the additional reference levels comprises multiple reference level segments, wherein each of the multiple reference level segments comprises a predetermined number of previous reference level segments, wherein generating each of the additional reference level segment hashes comprises performing an additional hashing function on the additional reference level segment hashes of each of the predetermined number of the previous reference level segments.

7. The computerized method of claim 6, wherein a first received data item includes genesis data, wherein a first hash of each of the first reference level segments is a hash of an immediately previous first reference level segment except for a first hash of a first segment of the first reference level, wherein a first hash for each of the additional reference level segments is a hash of an immediately previous additional reference level segment except for a first hash of a first additional reference level segment of each of the additional reference levels.

8. The computerized method of claim 1, 2, 3, 4, 5, or 6, wherein generating a first reference level segment hash for each of the first reference level segments is performed regardless of whether any of the data items are received during the time interval, wherein when no data items are received during one of the first reference level segments, a placeholder first reference level segment hash is performed.

9. A non-transitory computer-readable storage medium including a set of instructions that, when executed by one or more processors, cause a machine to:

generate reference levels according to time intervals, wherein the first reference level comprises a predetermined number of the time intervals, wherein each of the time intervals of the remaining reference levels is comprised of a predetermined number of the time intervals of a previous reference level;

create hashes of data at the first reference level by performing a hashing function on the data according to the time interval in which the data is received;

generate first reference level time interval hashes by performing the hashing function on the hashes of the data at each of the time intervals of the first reference level up to the predetermined number of the time intervals of the first reference level; and

generate reference level time interval hashes for the remaining reference levels by performing the hashing function on the hashes of the each of the time intervals of the previous reference level up to the predetermined number of the time intervals.

10. The non-transitory computer-readable storage medium of claim 9, wherein the set of instructions, when executed by the one or more processors, further cause the machine to record at least one of the reference level time interval hashes to a distributed ledger.

11. The non-transitory computer-readable storage medium of claim 10, wherein the set of instructions, when executed by the one or more processors, further cause the machine to:

process received verification data spanning a time period of the at least one of the reference level time interval hashes;

generate the reference level time interval hashes for the verification data; and

compare the reference level time interval hash for the verification data with the at least one of the reference level hashes to verify that the verification data is unchanged from the data.

12. The non-transitory computer-readable storage medium of claim 9, 10, or 11, wherein the time interval of the first reference level is seconds, and wherein the predetermined number of the time intervals in the first reference level is sixty or sixty-one.

13. The non-transitory computer-readable storage medium of claim 12, wherein the remaining reference levels comprise a second reference level, wherein the time interval of the second reference level is minutes, wherein the predetermined number of the time intervals in the second reference level is sixty.

14. The non-transitory computer-readable storage medium of claim 13, wherein the remaining reference levels further comprise a third reference level, wherein the time interval of the third reference level is hours, wherein the predetermined number of the time intervals in the third reference level is twenty-four.

15. The non-transitory computer-readable storage medium of claim 10, 11, 12, 13, or 14, wherein the set of instructions, when executed by the one or more processors, further cause the machine to:

receive a cryptographic signature attesting to the data; and

record at least one of the additional reference level time interval hashes to a distributed ledger.

16. The non-transitory computer-readable storage medium of claim 9, 10, 11, 12, 13, 14, or 15, wherein the set of instructions, when executed by the one or more processors, further cause the machine to timestamp the data, wherein creating hashes of data by performing the hashing function on the data at the first reference level includes performing the hashing function on the data with the timestamp.

17. The non-transitory computer-readable storage medium of claim 9, 10, 11, 12, 13, 14, 15, or 16 wherein a first received data of the data includes genesis data, wherein a first hash of each of the time intervals of the first reference level is a hash of an immediately previous time interval except for a first hash of a first time interval of the first reference level time interval, wherein a first hash of each of the time intervals of each of the remaining reference levels is a hash of an immediately previous time interval except for a first hash of a first reference level time interval of the each of the remaining reference levels.

18. The non-transitory computer-readable storage medium of claim 9, 10, 11, 12, 13, 14, 15, 16, or 17, wherein the first reference level time interval hashes are performed regardless of whether the data was received during the time interval, wherein when no data is received during the time interval, a placeholder first reference level time interval hash is performed.

19. A data storage and verification platform, comprising:

one or more processors; and

a computer readable storage medium having instructions stored thereon, which when executed by the one or more processors cause the data storage and verification platform to:

generate reference levels according to time intervals, wherein the first reference level comprises a predetermined number of the time intervals, wherein each of the time intervals of the remaining reference levels is comprised of a predetermined number of the time intervals of a previous reference level;

create hashes of data at the first reference level by performing a hashing function on the data according to the time interval in which the data is received, wherein the data is hashed in an order it was received;

generate first reference level time interval hashes by combining the hashes of each of the data received during each of the time intervals in a temporal order and performing the hashing function on the combined hashes of the data at each of the time intervals of the first reference level up to the predetermined number of the time intervals of the first reference level,

wherein when the data is not received during any one of the time intervals of the first reference level, a placeholder hash for a first reference level time interval hash is generated; and

generate reference level time interval hashes for the remaining reference levels by combining the hashes of each of the time intervals of the previous reference level up to the predetermined number of the time intervals in a temporal order and performing the hashing function on the combined hashes of the each of the time intervals of the previous reference level up to the predetermined number of the time intervals.

20. The data storage and verification platform of claim 19, wherein a first received data of the data includes genesis data, wherein a first hash of each of the time intervals of the first reference level is a hash of an immediately previous time interval except for a first hash of a first time interval of the first reference level time interval, wherein a first hash of each of the time intervals of each of the remaining reference levels is a hash of an immediately previous time interval except for a first hash of a first reference level time interval of the each of the remaining reference levels.

21. The data storage and verification platform of claim 19 or 20, wherein combining the hashes of each of the data received during each of the time intervals in a temporal order comprises concatenating the hashes of the each of the data.

<u>Computer System Overview</u>

[0074] Embodiments of the present disclosure include various steps and operations, which have been described above. A variety of these steps and operations may be performed by hardware components or may be embodied in machine-executable instructions, which may be used to cause a general-purpose or special-purpose processor programmed with the instructions to perform the steps. Alternatively, the steps may be performed by a combination of hardware, software, and/or firmware. As such, Fig. 7 is an example of a computer system 700 with which embodiments of the present disclosure may be utilized. According to the present example, the computer system 700 includes an interconnect 710, at least one processor 720, at least one communication port 730, a main memory 740, a removable storage media 750, a read-only memory 760, and a mass storage device 770.

[0075] Processor(s) 720 can be any known processor. Communication port(s) 730 can be or include, for example, any of an RS-232 port for use with a modem-based dialup connection, a 10/100 Ethernet port, or a Gigabit port using copper or fiber. The nature of communication port(s) 730 may be chosen depending on a network such as a Local Area Network (LAN), Wide Area Network (WAN), or any network to which the computer system 700 connects.

[0076] Main memory 740 can be Random Access Memory (RAM), or any other dynamic storage device(s) commonly known in the art. Read-only memory 760 can be any static storage device(s) such as Programmable Read-Only Memory (PROM) chips for storing static information such as instructions for processor 720.

[0077] Mass storage device 770 can be used to store information and instructions. For example, hard disks such as the Adaptec® family of SCSI drives, an optical disc, an array of disks, such as the Adaptec family of RAID drives, or any other mass storage devices may be used.

[0078] Interconnect 710 can be or include one or more buses, bridges, controllers, adapters, and/or point-to-point connections. Interconnect 710 communicatively couples processor(s) 720 with the other memory, storage, and communication blocks. Interconnect 710 can be a PCI/PCI-X–based or an SCSI-based system bus, depending on the storage devices used.

[0079] Removable storage media 750 can be any kind of external hard drives, floppy drives, compact disc–read-only memory (CD-ROM), compact disc–rewritable (CD-RW), or digital video disc–read-only memory (DVD-ROM).

[0080] The components described above are meant to exemplify some types of possibilities. In no way should the aforementioned examples limit the disclosure, as they are only exemplary embodiments.

<u>Terminology</u>

[0081] Brief definitions of terms, abbreviations, and phrases used throughout this application are given below.

[0082] The terms "connected" or "coupled" and related terms are used in an operational sense and are not necessarily limited to a direct physical connection or coupling. Thus, for example, two devices may be coupled directly, or via one or more intermediary media or devices. As another example, devices may be coupled in such a way that information can be passed therebetween, while not sharing any physical connection with one another. Based on the disclosure provided herein, one of ordinary skill in the art will appreciate a variety of ways in which connection or coupling exists in accordance with the aforementioned definition.

[0083] The phrases "in some embodiments," "according to some embodiments," "in the embodiments shown," "in other embodiments," "embodiments," and the like generally mean that the particular feature, structure, or characteristic following the phrase is included in at least one embodiment of the present disclosure, and may be included in more than one embodiment of the present disclosure. In addition, such phrases do not necessarily refer to the same embodiment or different embodiments.

[0084] If the specification states that a component or feature "may," "can," "could," or "might" be included or have a characteristic, that particular component or feature is not required to be included or have the characteristic.

[0085] The term "responsive" includes completely or partially responsive.

[0086] The term "module" refers broadly to a software, hardware, or firmware (or any combination thereof) component. Modules are typically functional components that can generate useful data or other output using specified input(s). A module may or may not be self-contained. An application program (also called an "application") may include one or more modules, or a module can include one or more application programs.

[0087] The term "network" generally refers to a group of interconnected devices capable of exchanging information. A network may be as few as several personal computers on a Local Area Network (LAN) or as large as the Internet, a worldwide network of computers. As used herein, "network" is intended to encompass any network capable of transmitting information from one entity to another. In some cases, a network may be comprised of multiple networks, even multiple heterogeneous networks, such as one or more border networks, voice networks, broadband networks, financial networks, service provider networks, Internet Service Provider (ISP) networks, and/or Public Switched Telephone Networks (PSTNs), interconnected via gateways operable to facilitate communications between and among the various networks.

[0088] Also, for the sake of illustration, various embodiments of the present disclosure have herein been described in the context of computer programs, physical components, and logical interactions within modern computer networks. Importantly, while these embodiments describe various embodiments of the present disclosure in relation to modern computer networks and programs, the method and apparatus described herein are equally applicable to other systems, devices, and networks, as one skilled in the art will appreciate. As such, the illustrated applications of the embodiments of the present disclosure are not meant to be limiting, but instead are examples. Other systems, devices, and networks to which embodiments of the present disclosure are applicable include, for example, other types of communication and computer devices and systems. More specifically, embodiments are applicable to communication systems, services, and devices such as cell phone networks and compatible devices. In addition, embodiments are applicable to all levels of computing, from the personal computer to large network mainframes and servers.

[0089] In conclusion, the present disclosure provides novel systems, methods, and arrangements for storing and verifying data. While detailed descriptions of one or more embodiments of the disclosure have been given above, various alternatives, modifications, and equivalents will be apparent to those skilled in the art without varying from the spirit of the disclosure. For example, while the embodiments described above refer to particular features, the scope of this disclosure also includes embodiments having different combinations of features and embodiments that do not include all of the described features. Accordingly, the scope of the present disclosure is intended to embrace all such alternatives, modifications, and variations as fall within the scope of the claims, together with all equivalents thereof. Therefore, the above description should not be taken as limiting.

CLAIMS

What is claimed is

1. A computerized method comprising:

   receiving data items during first reference level segments of a first reference level;

   performing a hashing function on each of the data items with a corresponding timestamp;

   generating a first reference level segment hash for each of the first reference level segments by performing a second hashing function on the hashes of the data items in each of the first reference level segments according to their respective timestamps; and

   generating a second reference level segment hash for each of multiple second reference level segments, wherein the each of the multiple second reference level segments is comprised of a predetermined number of the first reference level segments, wherein the each of the multiple second reference level segment hashes are generated by performing a third hashing function on the predetermined number of the first reference level segment hashes.

2. The computerized method of claim 1, further comprising recording at least one of the second reference level segment hashes to a distributed ledger.

3. The computerized method of claim 1, wherein the first reference level segments and the second reference level segments are periods of time.

4. The computerized method of claim 3, wherein the period of time of the first reference level segment is a second, and wherein the period of time of the second reference level segment is a minute.

5. The computerized method of claim 4, wherein the predetermined number of the first reference level segments is sixty or sixty-one.

6. The computerized method of claim 1, further comprising generating reference level segment hashes for additional reference levels, wherein each of the additional reference levels comprises multiple reference level segments, wherein each of the multiple reference level segments comprises a predetermined number of previous reference level segments, wherein generating each of the additional reference level segment hashes comprises performing an additional hashing function on the additional reference level segment hashes of each of the predetermined number of the previous reference level segments.

7. The computerized method of claim 6, wherein a first received data item includes genesis data, wherein a first hash of each of the first reference level segments is a hash of an immediately previous first reference level segment except for a first hash of a first segment of the first reference level, wherein a first hash for each of the additional reference level segments is a hash of an immediately previous additional reference level segment except for a first hash of a first additional reference level segment of each of the additional reference levels.

8. The computerized method of claim 1, wherein generating a first reference level segment hash for each of the first reference level segments is performed regardless of whether any of the data items are received during the time interval, wherein when no data items are received during one of the first reference level segments, a placeholder first reference level segment hash is performed.

9. A non-transitory computer-readable storage medium including a set of instructions that, when executed by one or more processors, cause a machine to:

generate reference levels according to time intervals, wherein the first reference level comprises a predetermined number of the time intervals, wherein each of the time intervals of remaining reference levels is comprised of a predetermined number of the time intervals of a previous reference level;

create hashes of data at the first reference level by performing a hashing function on the data according to the time interval in which the data is received;

generate first reference level time interval hashes by performing the hashing function on the hashes of the data at each of the time intervals of the first reference level up to the predetermined number of the time intervals of the first reference level; and

generate reference level time interval hashes for the remaining reference levels by performing the hashing function on the hashes of the each of the time intervals of the previous reference level up to the predetermined number of the time intervals.

10. The non-transitory computer-readable storage medium of claim 9, wherein the set of instructions, when executed by the one or more processors, further cause the machine to record at least one of the reference level time interval hashes to a distributed ledger.

11. The non-transitory computer-readable storage medium of claim 10, wherein the set of instructions, when executed by the one or more processors, further cause the machine to:

process received verification data spanning a time period of the at least one of the reference level time interval hashes;

generate the reference level time interval hashes for the verification data; and

compare the reference level time interval hash for the verification data with the at least one of the reference level hashes to verify that the verification data is unchanged from the data.

12. The non-transitory computer-readable storage medium of claim 9, wherein the time interval of the first reference level is seconds, and wherein the predetermined number of the time intervals in the first reference level is sixty or sixty-one.

13. The non-transitory computer-readable storage medium of claim 12, wherein the remaining reference levels comprise a second reference level, wherein the time interval of the second reference level is minutes, wherein the predetermined number of the time intervals in the second reference level is sixty.

14. The non-transitory computer-readable storage medium of claim 13, wherein the remaining reference levels further comprise a third reference level, wherein the time interval of the third reference level is hours, wherein the predetermined number of the time intervals in the third reference level is twenty-four.

15. The non-transitory computer-readable storage medium of claim 10, wherein the set of instructions, when executed by the one or more processors, further cause the machine to:

receive a cryptographic signature attesting to the data; and

record at least one of the additional reference level time interval hashes to a distributed ledger.

16. The non-transitory computer-readable storage medium of claim 9, wherein the set of instructions, when executed by the one or more processors, further cause the machine to timestamp the data, wherein creating hashes of data by performing the hashing function on the data at the first reference level includes performing the hashing function on the data with the timestamp.

17. The non-transitory computer-readable storage medium of claim 9, wherein a first received data of the data includes genesis data, wherein a first hash of each of the time intervals of the first reference level is a hash of an immediately previous time interval except for a first hash of a first time interval of the first reference level time interval, wherein a first hash of each of the time intervals of each of the remaining reference levels is a hash of an immediately previous time interval except for a first hash of a first reference level time interval of the each of the remaining reference levels.

18. The non-transitory computer-readable storage medium of claim 9, wherein the first reference level time interval hashes are performed regardless of whether the data was received during the time interval, wherein when no data is received during the time interval, a placeholder first reference level time interval hash is performed.

19. A data storage and verification platform, comprising:

one or more processors; and

a computer readable storage medium having instructions stored thereon, which when executed by the one or more processors cause the data storage and verification platform to:

generate reference levels according to time intervals, wherein the first reference level comprises a predetermined number of the time intervals, wherein each of the time intervals of the remaining reference levels is comprised of a predetermined number of the time intervals of a previous reference level;

create hashes of data at the first reference level by performing a hashing function on the data according to the time interval in which the data is received, wherein the data is hashed in an order it was received;

generate first reference level time interval hashes by combining the hashes of each of the data received during each of the time intervals in a temporal order and performing the hashing function on the combined hashes of the data at each of the time intervals of the first reference level up to the predetermined number of the time intervals of the first reference level,

wherein when the data is not received during any one of the time intervals of the first reference level, a placeholder hash for a first reference level time interval hash is generated; and

generate reference level time interval hashes for the remaining reference levels by combining the hashes of each of the time intervals of the previous reference level up to the predetermined number of the time intervals in a temporal order and performing the hashing function on the combined hashes of the each of the time intervals of the previous reference level up to the predetermined number of the time intervals.

20. The data storage and verification platform of claim 19, wherein a first received data of the data includes genesis data, wherein a first hash of each of the time intervals of the first reference level is a hash of an immediately previous time interval except for a first hash of a first time interval of the first reference level time interval, wherein a first hash of each of the time intervals of each of the remaining reference levels is a hash of an immediately previous time interval except for a first hash of a first reference level time interval of the each of the remaining reference levels.

21. The data storage and verification platform of claim 19, wherein combining the hashes of each of the data received during each of the time intervals in a temporal order comprises concatenating the hashes of the each of the data.

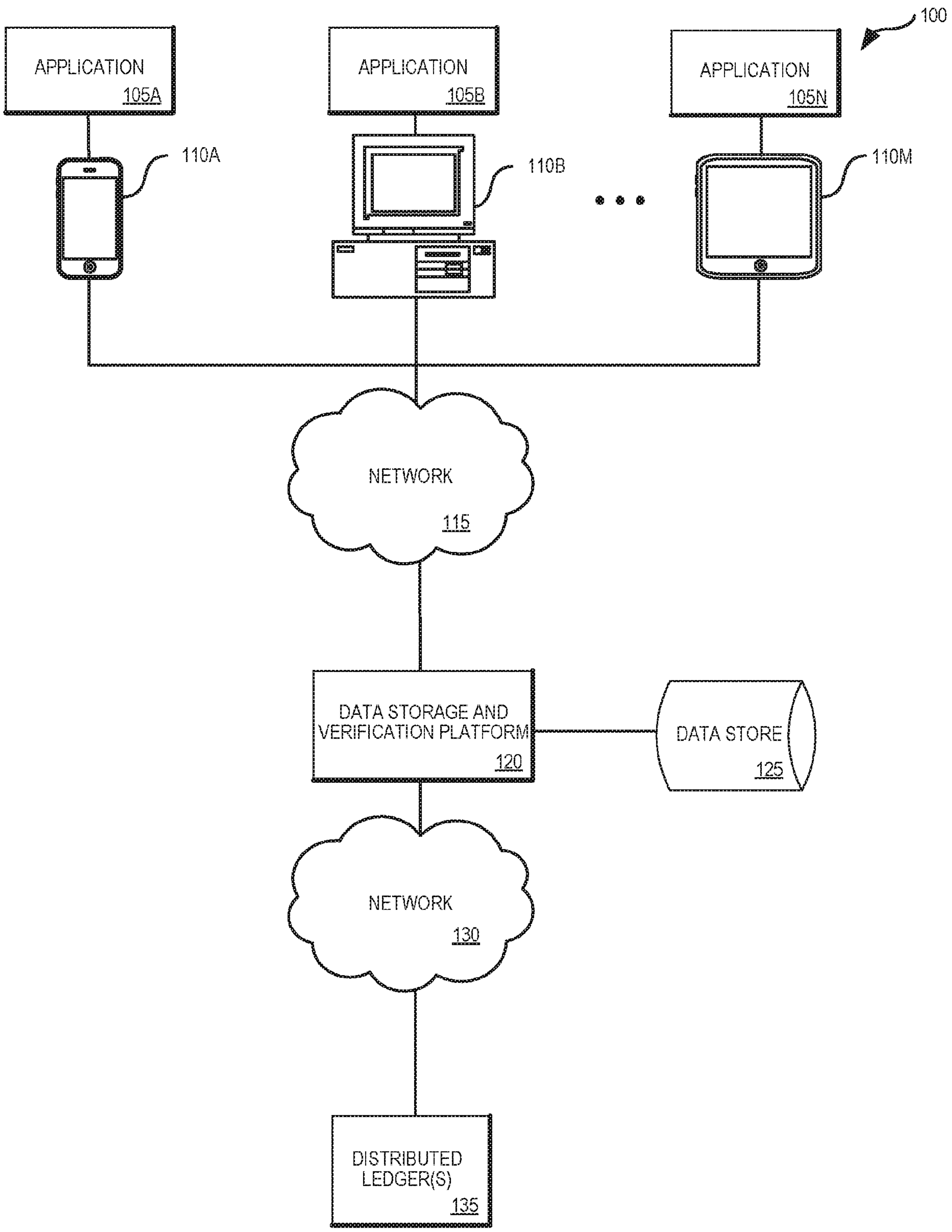

100
APPLICATION 105A
APPLICATION 105B
APPLICATION 105N
110A
110B
110M
NETWORK 115
DATA STORAGE AND VERIFICATION PLATFORM 120
DATA STORE 125
NETWORK 130
DISTRIBUTED LEDGER(S) 135

FIG. 1

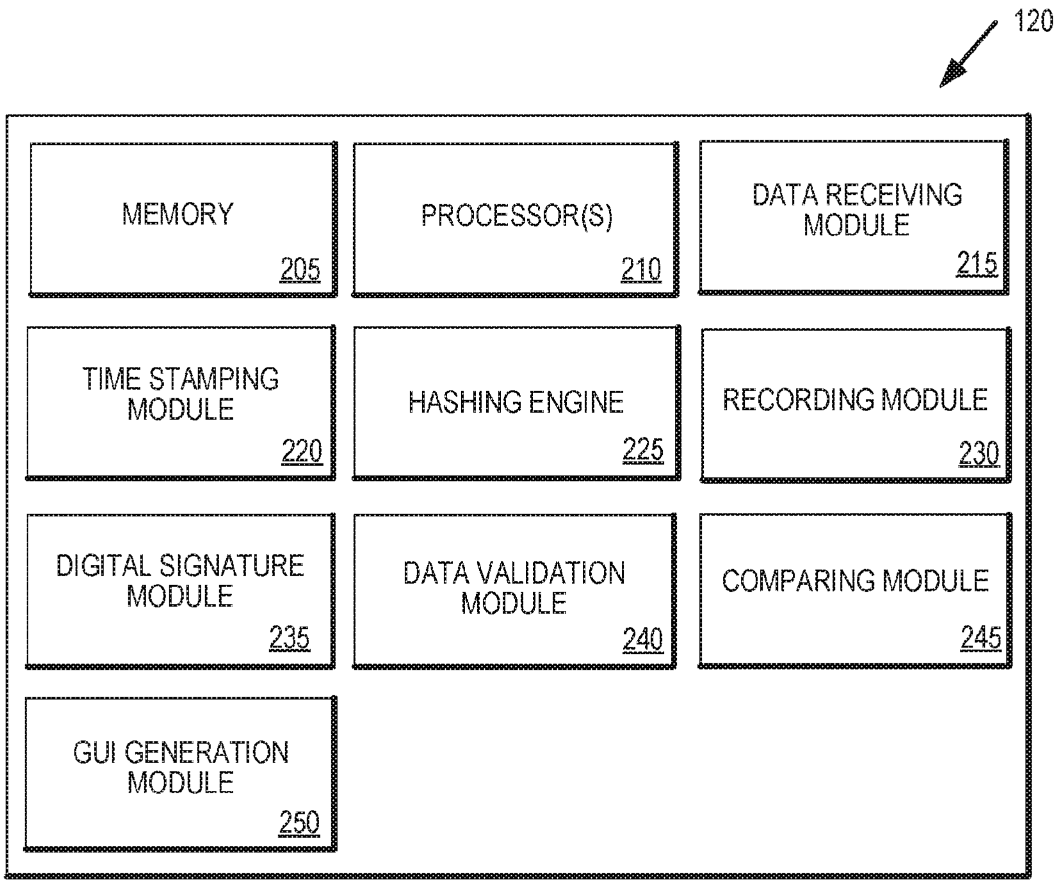

120
MEMORY
205
PROCESSOR(S)
210
DATA RECEIVING MODULE
215
TIME STAMPING MODULE
220
HASHING ENGINE
225
RECORDING MODULE
230
DIGITAL SIGNATURE MODULE
235
DATA VALIDATION MODULE
240
COMPARING MODULE
245
GUI GENERATION MODULE
250

FIG. 2

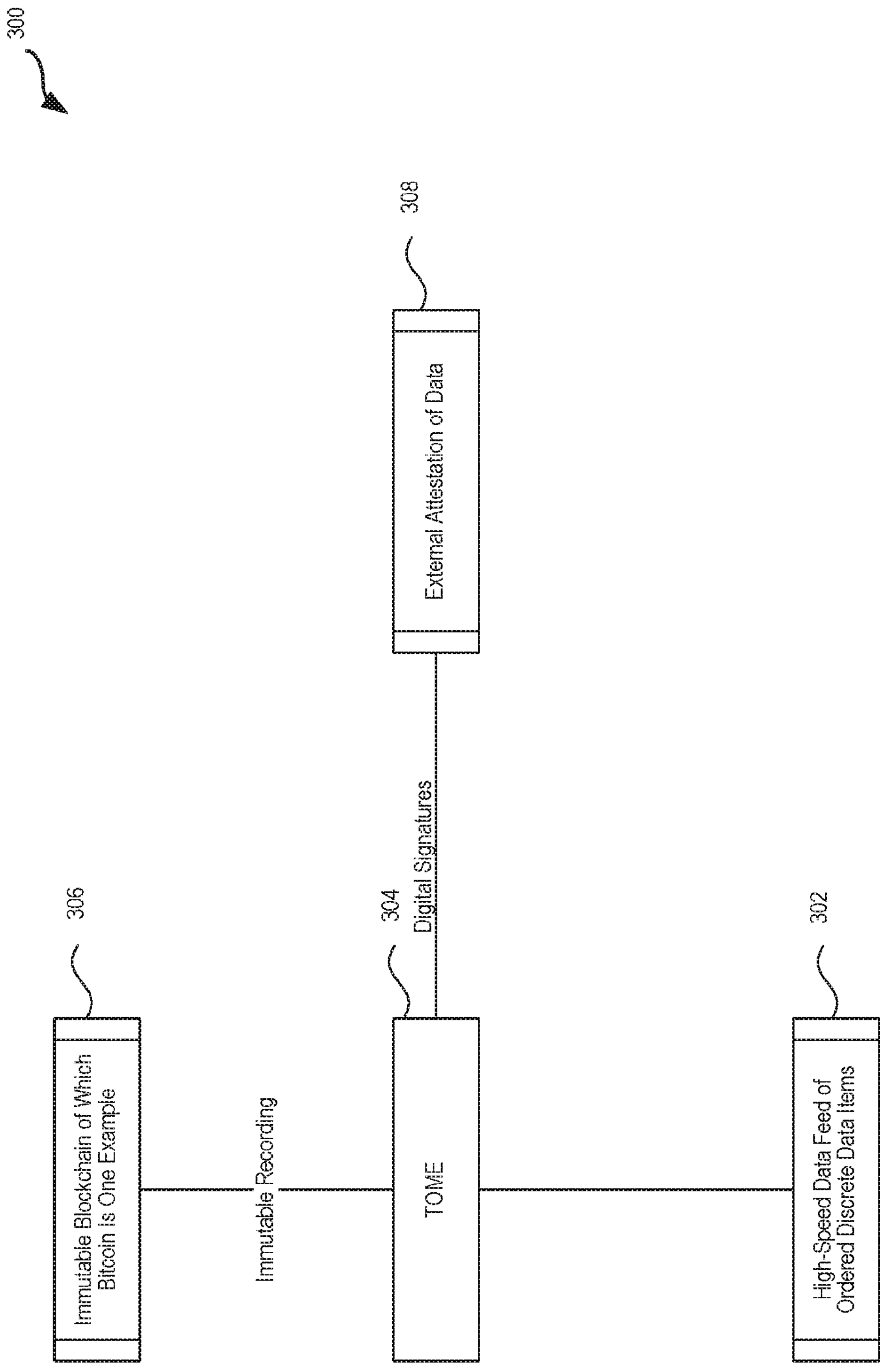

300
308
External Attestation of Data
Digital Signatures
306
Immutable Blockchain of Which Bitcoin is One Example
Immutable Recording
304
TOME
302
High-Speed Data Feed of Ordered Discrete Data Items

FIG. 3

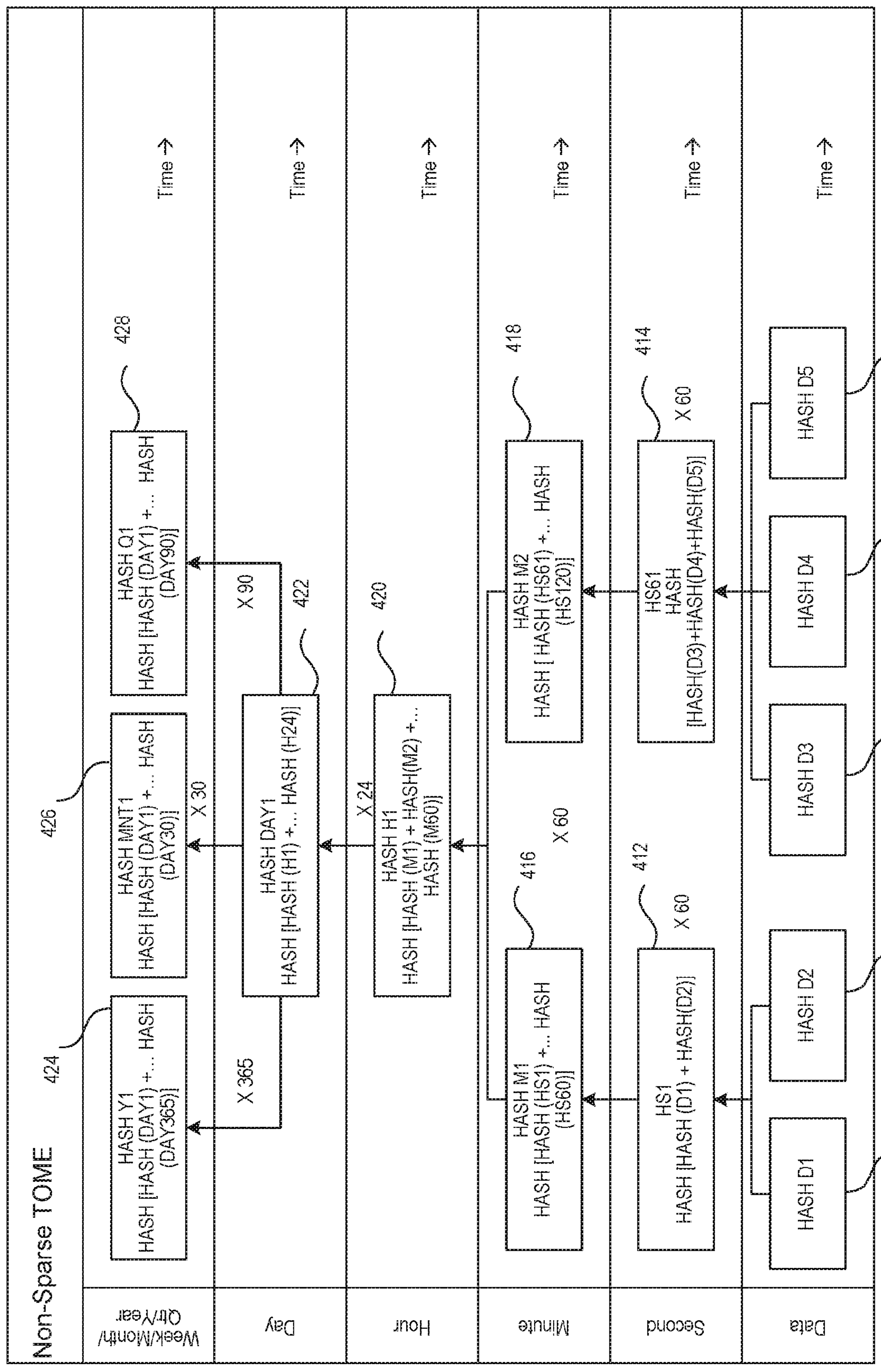

Non-Sparse TOME
Week/Month/Qtr/Year
Day
Hour
Minute
Second
Data
Time →
HASH Y1 HASH [HASH (DAY1) +... HASH (DAY365)]
424
HASH MNT1 HASH [HASH (DAY1) +... HASH (DAY30)]
426
HASH Q1 HASH [HASH (DAY1) +... HASH (DAY90)]
428
X 365
X 30
X 90
HASH DAY1 HASH [HASH (H1) +... HASH (H24)]
422
X 24
HASH H1 HASH [HASH (M1) + HASH(M2) +... HASH (M60)]
420
HASH M1 HASH [HASH (HS1) +... HASH (HS60)]
416
X 60
HASH M2 HASH [HASH (HS61) +... HASH (HS120)]
418
HS1 HASH [HASH (D1) + HASH(D2)]
412
X 60
HS61 HASH [HASH(D3)+HASH(D4)+HASH(D5)]
414
X 60
HASH D1
402
HASH D2
404
HASH D3
406
HASH D4
408
HASH D5
410

FIG. 4

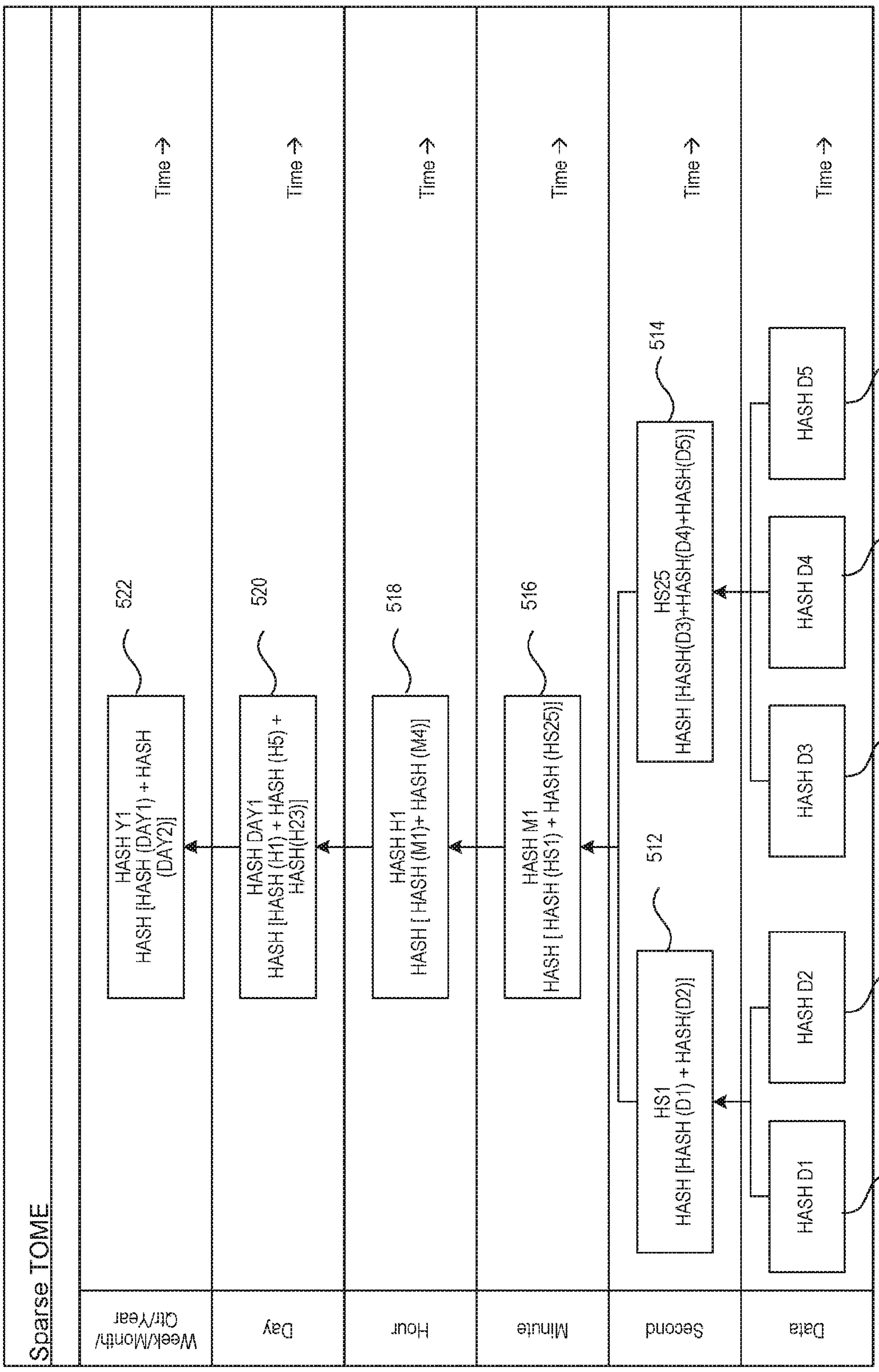

Sparse TOME
Week/Month/Qtr/Year
Day
Hour
Minute
Second
Data
Time →
Time →
Time →
Time →
Time →
Time →
HASH Y1 HASH [HASH (DAY1) + HASH (DAY2)]
522
HASH DAY1 HASH [HASH (H1) + HASH (H5) + HASH(H23)]
520
HASH H1 HASH [HASH (M1)+ HASH (M4)]
518
HASH M1 HASH [HASH (HS1) + HASH (HS25)]
516
HS1 HASH [HASH (D1) + HASH(D2)]
512
HS25 HASH [HASH(D3)+HASH(D4)+HASH(D5)]
514
HASH D1
502
HASH D2
504
HASH D3
506
HASH D4
508
HASH D5
510

FIG. 5

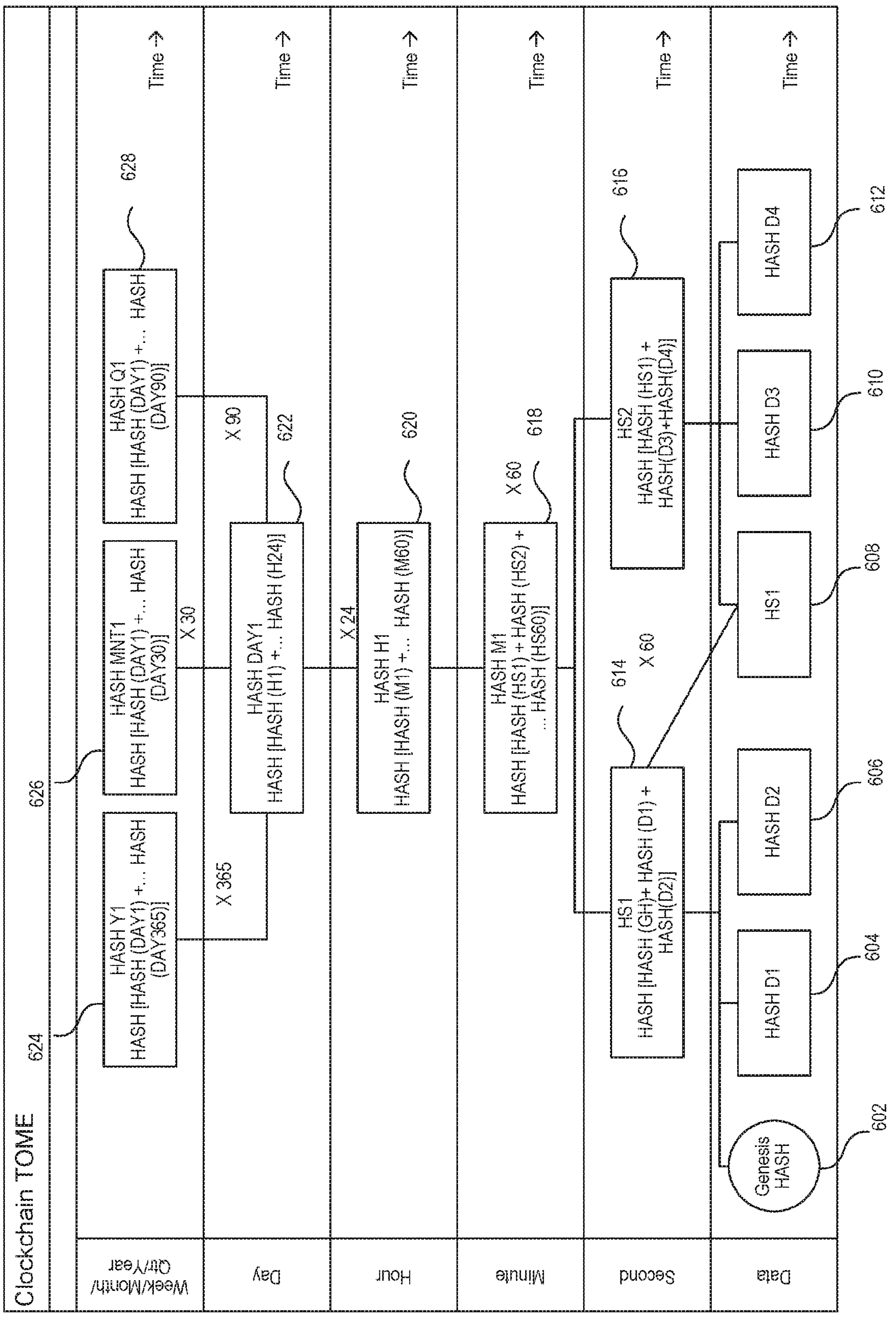

Clockchain TOME
Week/Month/Qtr/Year
Day
Hour
Minute
Second
Data
Time →
Genesis HASH
602
HASH D1
604
HASH D2
606
HS1
608
HASH D3
610
HASH D4
612
HS1 HASH [HASH (GH)+ HASH (D1) + HASH(D2)]
614
X 60
HS2 HASH [HASH (HS1) + HASH(D3)+HASH(D4)]
616
HASH M1 HASH [HASH (HS1) + HASH (HS2) + ... HASH (HS60)]
618
X 60
HASH H1 HASH [HASH (M1) +... HASH (M60)]
620
X 24
HASH DAY1 HASH [HASH (H1) +... HASH (H24)]
622
X 365
X 30
X 90
HASH Y1 HASH [HASH (DAY1) +... HASH (DAY365)]
624
HASH MNT1 HASH [HASH (DAY1) +... HASH (DAY30)]
626
HASH Q1 HASH [HASH (DAY1) +... HASH (DAY90)]
628

FIG. 6

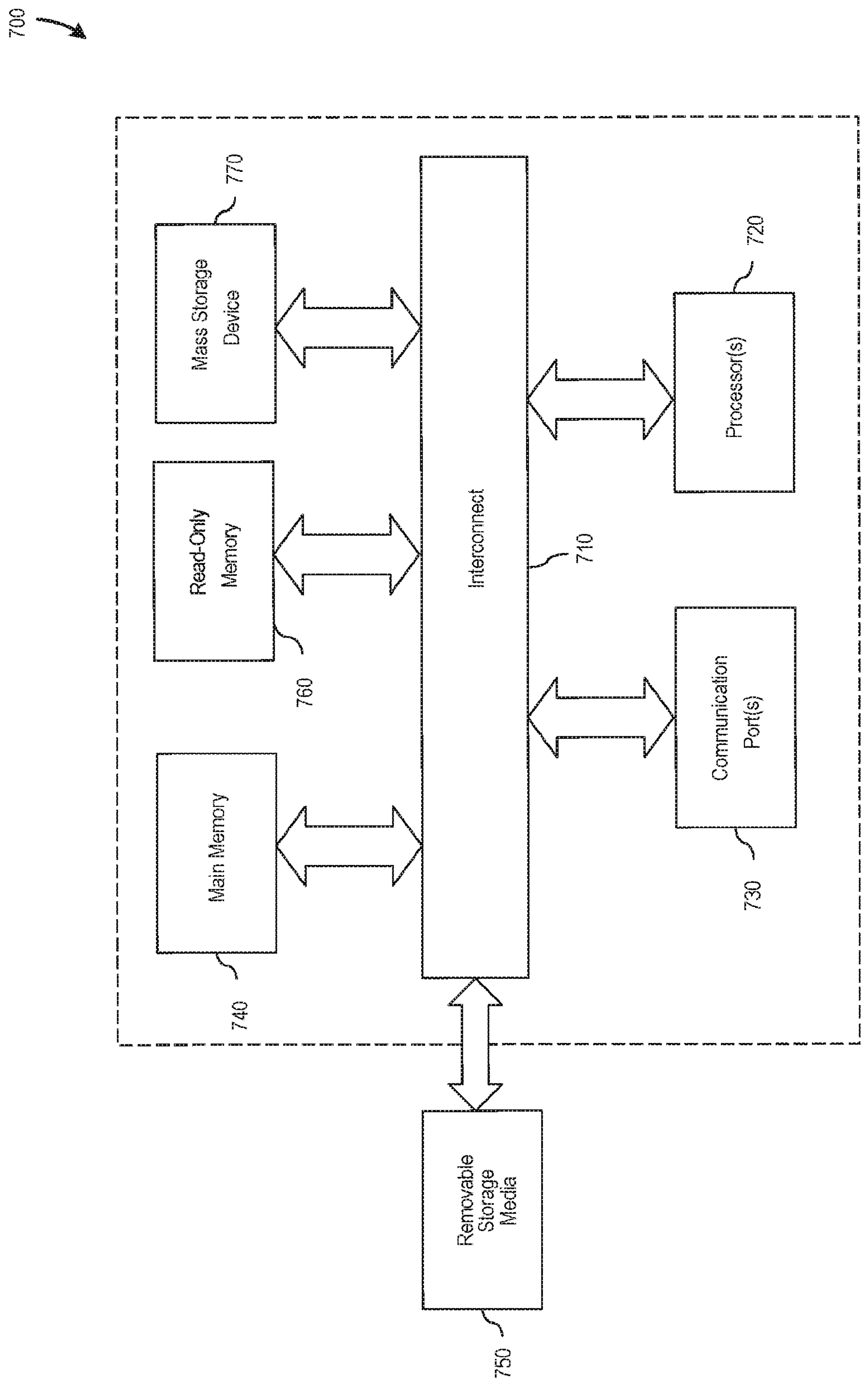

700
Mass Storage Device
770
Read-Only Memory
760
Main Memory
740
Interconnect
710
Processor(s)
720
Communication Port(s)
730
Removable Storage Media
750

FIG. 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/051301

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06F 21/64; G06Q 30/02; H04L 9/08 (2016.01)
CPC - G10L 19/0019; H04K 1/00; H04K 1/04 (2016.08)
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC - G06F 21/64; G06Q 30/02; H04L 9/08
CPC - G10L 19/0019; H04K 1/00; H04K 1/04

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
USPC - 380/44; 380/282; 713/175 (keyword delimited)

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
Orbit, Google Patents, Google
Search terms used: timestamp, merkel, tree, block, chain, hash, data, verificaiton

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X — Y	GASSEND et al. "Caches and Merkle Trees for Efficient Memory Authentication," Proceedings of the 9th High Performance Computer Architecture Symposium (HPCA'03) 2002, September, pp 1-14, [retrieved on 2016-11-01]. Retrieved from the Internet: <URL: http://csg.csail.mit.edu/pubs/memos/Memo-453/memo-453.pdf>	1, 6, 7 — 2-5, 8
X — Y	US 2005/0166263 A1 (NANOPOULOS et al) 28 July 2005 (28.07.2005) entire document	9, 12-14, 18, 19, 21 — 3-5, 8, 10, 11, 15-17, 20
Y	NAIK. "Optimising the SHA256 Hashing Algorithm for Faster and More Efficient Bitcoin Mining," MSc Information Security Department of Computer Science, University College of London, 02 September 2013, pp 1-65 [retrieved on 2016-11-01]. Retrieved from the Internet: <URL: http://www.nicolascourtois.com/bitcoin/Optimising%20the%20SHA256%20Hashing% 20Algorithm%20for%20Faster%20and%20More%20Efficient%20Bitcoin% 20Mining_Rahul_Naik.pdf>	2, 10, 11, 15-17, 20
A	US 2010/0212017 A1 (LI et al) 19 August 2010 (19.08.2010) entire document	1-21

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

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

Date of the actual completion of the international search 01 November 2016	Date of mailing of the international search report 22 NOV 2016
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, VA 22313-1450 Facsimile No. 571-273-8300	Authorized officer Blaine R. Copenheaver PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 108292351 A
(43)申请公布日 2018.07.17

(21)申请号 201680066334.6
(22)申请日 2016.09.12
(30)优先权数据
14/852,955 2015.09.14 US
(85)PCT国际申请进入国家阶段日
2018.05.14
(86)PCT国际申请的申请数据
PCT/US2016/051301 2016.09.12
(87)PCT国际申请的公布数据
WO2017/048630 EN 2017.03.23
(71)申请人 缇零网股份有限公司
地址 美国犹他州
(72)发明人 T·布莱克 A·威尔金斯
R·克里斯顿森
(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 姬利永 钱慰民
(51)Int.Cl.
G06F 21/64(2006.01)
G06Q 30/02(2006.01)
H04L 9/08(2006.01)

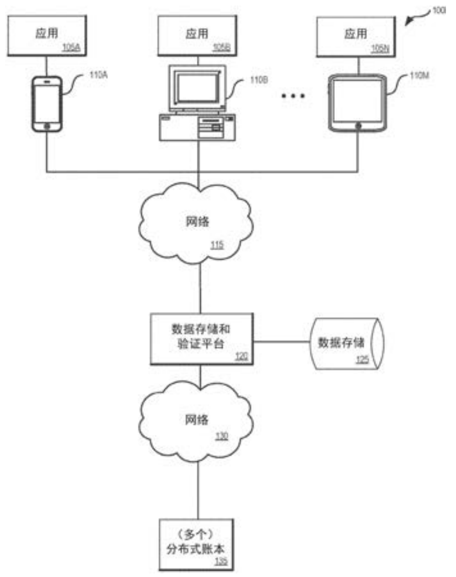
权利要求书3页 说明书15页 附图7页

(54)发明名称

使用诸如以时间为中心的默克尔散列树之类的散列树的数据验证方法和系统

(57)摘要

本文描述的系统和方法通常涉及存储和验证数据。在一些实施例中，根据时间间隔生成参考级，其中所述第一参考级包括预定数量的时间间隔，并且其中剩余参考级的时间间隔中的每个由先前的参考级的预定数量的时间间隔组成。通过以时间顺序的方式对数据执行散列函数可以在第一参考级创建数据的散列。通过对在第一参考级的时间间隔中的每个处的数据的散列执行散列函数可以生成第一参考级时间间隔散列。通过对先前的参考级的时间间隔中的每个的散列执行散列函数可以生成针对剩余参考级时间间隔的散列。



1.一种计算机化方法,包括:

在第一参考级的第一参考级段期间接收数据项;

用对应的时间戳对所述数据项中的每个执行散列函数;

通过对所述第一参考级段中的每个中的所述数据项的散列根据他们各自的时间戳执行第二散列函数来生成针对所述第一参考级段中的每个的第一参考级段散列;以及

生成针对多个第二参考级段中的每个的第二参考级段散列,其中所述多个第二参考级段中的每个由预定数量的所述第一参考级段组成,其中所述多个第二参考级段散列中的每个通过对预定数量的所述第一参考级段散列执行第三散列函数来生成。

2.根据权利要求1所述的计算机化方法,进一步包括将所述第二参考级段散列中的至少一个记录到分布式账本。

3.根据权利要求1所述的计算机化方法,其中所述第一参考级段和所述第二参考级段是时间段。

4.根据权利要求3所述的计算机化方法,其中所述第一参考级段的时间段是秒,并且其中所述第二参考级段的时间段是分钟。

5.根据权利要求4所述的计算机化方法,其中所述第一参考级段的所述预定数量是六十或六十一。

6.根据权利要求1所述的计算机化方法,进一步包括针对附加参考级生成参考级段散列,其中所述附加参考级中的每个包括多个参考级段,其中所述多个参考级段中的每个包括预定数量的先前的参考级段,其中生成所述附加参考级段散列中的每个包括对所述预定数量的所述先前的参考级段中的每个的所述附加参考级段散列执行附加散列函数。

7.根据权利要求6所述的计算机化方法,其中第一接收数据项包括创世数据,其中所述第一参考级段中的每个的第一散列是除了所述第一参考级的第一段的第一散列之外的紧接先前的第一参考级段的散列,其中针对所述附加参考级段中的每个的第一散列是除了所述附加参考级中的每个的第一附加参考级段的第一散列之外的紧接先前的附加参考级段的散列。

8.根据权利要求1所述的计算机化方法,其中执行生成针对所述第一参考级段中的每个的第一参考级段散列,而不管在所述时间间隔期间是否接收到所述数据项中的任何数据项,其中当在第一参考级段中的一个期间没有接收到数据项时,执行占位符第一参考级段散列。

9.一种非瞬态计算机可读存储介质,包括一组指令,所述一组指令在由一个或多个处理器执行时使机器用于:

根据时间间隔生成参考级,其中第一参考级包括预定数量的时间间隔,其中剩余参考级的所述时间间隔中的每个由先前的参考级的预定数量的时间间隔组成;

通过根据其中接收到数据的时间间隔对所述数据执行散列函数来创建在第一参考级处的数据的散列;

通过对在第一参考级的所述时间间隔中的每个处的数据的散列执行散列函数,直到第一参考级的所述预定数量的时间间隔,来生成第一参考级时间间隔散列;以及

通过对所述先前的参考级的所述时间间隔中的每个的散列执行散列函数直到所述预定数量的时间间隔,来生成针对所述剩余参考级的参考级时间间隔散列。

10.根据权利要求9所述的非瞬态计算机可读存储介质，其中在由所述一个或多个处理器执行时，所述一组指令进一步使所述机器将所述参考级时间间隔散列中的至少一个记录到分布式账本。

11.根据权利要求10所述的非瞬态计算机可读存储介质，其中所述一组指令在由所述一个或多个处理器执行时进一步使所述机器用于：

处理接收到的验证数据，所述验证数据跨越所述参考级时间间隔散列中的所述至少一个的时间段；

生成针对所述验证数据的所述参考级时间间隔散列；以及

将针对所述验证数据的所述参考级时间间隔散列与所述参考级散列中的所述至少一个进行比较，以验证所述验证数据从所述数据没有变化。

12.根据权利要求9所述的非瞬态计算机可读存储介质，其中所述第一参考级的时间间隔是秒，并且其中在所述第一参考级中的时间间隔的所述预定数量是六十或六十一。

13.根据权利要求12所述的非瞬态计算机可读存储介质，其中所述剩余参考级包括第二参考级，其中所述第二参考级的时间间隔是分钟，其中所述第二参考级中的时间间隔的所述预定数量是六十。

14.根据权利要求13所述的非瞬态计算机可读存储介质，其中所述剩余参考级进一步包括第三参考级，其中所述第三参考级的时间间隔是小时，其中所述第三参考级中的时间间隔的所述预定数量是二十四。

15.根据权利要求10所述的非瞬态计算机可读存储介质，其中当由所述一个或多个处理器执行时，所述一组指令进一步使所述机器用于：

接收证明所述数据的密码签名；以及

将所述附加参考级时间间隔散列中的至少一个记录到分布式账本。

16.根据权利要求9所述的非瞬态计算机可读存储介质，其中当由所述一个或多个处理器执行时，所述一组指令进一步使所述机器用于对所述数据加时间戳，其中通过对在第一参考级的数据执行散列函数来创建数据的散列包括用所述时间戳对数据执行散列函数。

17.根据权利要求9所述的非瞬态计算机可读存储介质，其中所述数据的第一接收数据包括创世数据，其中所述第一参考级的时间间隔中的每个的第一散列是除了所述第一参考级时间间隔的第一时间间隔的第一散列之外的紧接先前的时间间隔的散列，其中所述剩余参考级中的每个的时间间隔中的每个的第一散列是除了所述剩余参考级中的每个的第一参考级时间间隔的第一散列之外的紧接先前的时间间隔的散列。

18.根据权利要求9所述的非瞬态计算机可读存储介质，其中执行所述第一参考级时间间隔散列，而不管在所述时间间隔期间是否接收到所述数据，其中当在所述时间间隔期间没有接收到数据时，执行占位符第一参考级时间间隔散列。

19.一种数据存储和验证平台，包括：

一个或多个处理器；以及

计算机可读存储介质，具有存储在其上的指令，所述指令当由所述一个或多个处理器执行时使所述数据存储和验证平台用于：

根据时间间隔生成参考级，其中所述第一参考级包括预定数量的时间间隔，其中剩余参考级的时间间隔中的每个由先前的参考级的预定数量的时间间隔组成；

通过根据其中接收到数据的时间间隔对所述数据执行散列函数来创建在第一参考级处的数据的散列,其中所述数据以其被接收的顺序散列;

通过以时间顺序组合在时间间隔中的每个期间接收到的数据中的每个的散列,并且对在所述第一参考级的时间间隔中的每个处的数据的被组合的散列执行散列函数直到所述第一参考级的预定数量的时间间隔,来生成第一参考级时间间隔散列,

其中当在所述第一参考级的时间间隔中的任何一个时间间隔期间没有接收到数据时,生成针对第一参考级时间间隔散列的占位符散列;以及

通过以时间顺序组合所述先前的参考级的时间间隔中的每个的散列直到所述预定数量的时间间隔并且对所述先前的参考级的时间间隔中的每个的被组合的散列执行散列函数直到所述预定数量的时间间隔来生成针对所述剩余参考级的参考级时间间隔散列。

20.根据权利要求19所述的数据存储和验证平台,其中所述数据的第一接收数据包括创世数据,其中所述第一参考级的时间间隔中的每个的第一散列是除了所述第一参考级时间间隔的第一时间间隔的第一散列之外的紧接先前的时间间隔的散列,其中所述剩余参考级中的每个的时间间隔中的每个的第一散列是除了所述剩余参考级中的每个的第一参考级时间间隔的第一散列之外的紧接先前的时间间隔的散列。

21.根据权利要求19所述的数据存储和验证平台,其中以时间顺序组合在时间间隔中的每个期间接收到的数据中的每个的散列包括连接所述数据中的每个的散列。

使用诸如以时间为中心的默克尔散列树之类的散列树的数据验证方法和系统

相关申请的交叉引用

[0001]　本申请要求于2015年9月14日提交的名称为“DATA VERIFICATION METHODS AND SYSTEMS USING A HASH TREE,SUCH AS A TIME-CENTRIC MERKLE HASH TREE(使用诸如时间中心的默克尔散列树之类的散列树的数据验证方法和系统)”的美国专利申请号14/852,955的优先权和权益,所述美国专利申请的全部内容出于所有目的通过引用以其全文结合在此。

技术领域

[0002]　本公开的各种实施例通常涉及存储和验证数据。更具体地,本公开的各种实施例涉及用于使用散列(hashing)技术存储和验证数据的系统和方法。

背景技术

[0003]　散列(hash)函数是一种可用于将任意大小的数字数据映射到固定大小的数字数据的函数。散列函数可用于许多目的,例如用于通过检测大文件中的复制记录来加速表或数据库的查找。散列函数也被用于区块链中。区块链是可验证的永久性账本,该永久性账本用被附接到每个区块并验证该区块的工作量证明密封(散列)来每次被构建一个区块。在任何区块链中,前一个区块的散列被包括在当前区块中,并且因此通过递归,当前散列也将所有先前的区块验证回到原始的创世(genesis)区块。将散列插入到区块链中永久地记录该散列并且充当在区块被添加到链中的时刻验证散列数据存在的时间戳证明的公证人。未来的区块从链重组添加了保护层,并因此添加了对链中较早的区块不能做出改变的确定性。

附图说明

[0004]　将通过使用附图来描述和解释本公开的实施例。

[0005]　图1图示了根据本公开的各种实施例的基于网络的操作环境的示例。

[0006]　图2图示了根据本公开的一个或多个实施例的数据存储和验证平台中的一组组件。

[0007]　图3是图示了根据本公开的一个或多个实施例的数据存储和验证平台的架构的图。

[0008]　图4图示了根据本公开的一个或多个实施例的使用非稀疏TOME来存储和验证数据的过程。

[0009]　图5图示了根据本公开的一个或多个实施例的使用稀疏TOME来存储和验证数据的过程。

[0010]　图6图示了根据本公开的一个或多个实施例的使用时钟链TOME来存储和验证数据的过程。

[0011]　图7图示了用其本公开的一些实施例可以被使用的计算机系统的示例。

具体实施方式

[0012] 本公开的各种实施例通常涉及存储和验证数据。更具体地，本公开的各种实施例涉及用于使用散列技术存储和验证数据的系统和方法。

[0013] 数据存储和验证平台描述了一种方法和系统，其中使用默克尔(Merkle)树(其中树分支是以时间为中心的数据分组)的概念可以既快速又高效地生成数据记录以及特别地快速变化的时间敏感的数据的记录。

[0014] 数据验证的传统方法是效率低下的，特别是在涉及大量快速变化的数据(例如，交易数据、遥测)的情况下。例如，用于验证数据的一种解决方案包括存储文件或文本的整个语料库并将其与原始数据进行比较以确认有效性。尽管这种方法对于小量数据是可管理的，但该方案对于任何显著量数据的比较都是不切实际的。

[0015] 另一个当前的解决方案是将数据存储在区块链中。但是，因为至少两个原因，在区块链中记录快速变化的数据(例如，在交易所中的每笔交易)是不切实际的。首先，当前必须同步和存储的数据量超过了当前区块链的大多数通信通道、当前区块链的大多数存储系统和当前区块链的“带宽”。简而言之，快速变化的数据不能跨广泛地分布的分散系统被快速同步。其次，在区块链上记录快速变化的数据是不切实际的，因为分散区块链的时序不是确定性的。也就是说，用于数据的路径取决于对等连接，这可能导致以与其最初发生的不同顺序记录快速变化的数据。

[0016] 本文描述的方法和系统提供了记录数据(包括快速变化的时间敏感的数据)以供将来验证的方式。本文描述的一些实施例描述了可以生成时间有序默克尔树时期(TOME)的数据存储和验证平台，其中“时间”可以指视为整体的过去、现在和将来中存在和事件的无限连续进展；“有序”可以指以有条不紊的或适当的方式进行安排；其中“默克尔树”指由Ralph Merkle发明的默克尔树，其中每个非叶节点用其子节点的标签的散列来标记；“时期(epoch)”指在某人或某事的历史中独特时期的开始。TOME可以使用诸如SHA256之类的加密散列函数用于散列。

[0017] 在一些实施例中，TOME的叶可以是特定记录，并且树可以由每第一参考级的数据记录的数量，第一参考级中组成第二参考级的段或时间间隔的数量，第二参考级中组成第三参考级的段或时间间隔的数量等等来定义。在一些实施例中，参考级由时间定义，并且在这样的实施例中，树可以由每秒的数据记录的数量、每分钟的秒数、每小时的分钟数、每日的小时数等来定义。

[0018] 在示例中并且如本文进一步描述的，如果需要，数据存储和验证平台可以接收数据记录并提供时间戳。在接收到数据记录时，每个数据记录可以用由数据存储和验证平台生成的对应的时间戳来散列。时间戳可以表示不同级的粒度(例如，皮秒、秒等)。在一个说明性的示例中，以秒来表示粒度，并且因此每个数据记录的散列以时间/相继的顺序被组合，并且每秒生成散列的散列；被组合的散列可以被称为一秒TOME。因此，每秒创建一个散列。在已经创建了六十个一秒散列之后，所有的一秒散列然后被组合(按时间上升)并且被散列用于创建一分钟散列，其可以被称为一分钟TOME。在已经创建了一分钟散列的六十个之后，所有的六十个一分钟散列被排序(按时间上升)，并被散列用于创建一小时TOME。在已经创建了一小时散列的二十四个之后，二十四个一小时散列被排序(按时间上升)，并被散

列用于生成一日TOME。该日散列然后可以被排序(按时间上升),并被散列用于产生一个月TOME(即三十个或三十一个每日散列的散列),季度TOME(即九十个每日散列的散列),和/或年度TOME(即365个每日散列的散列)。

[0019] 可以使用各种方法来完成在对被组合的散列执行散列函数之前组合散列。例如,可以通过将散列连接在一起,通过数字上相加散列,或通过将二进制字节连接在一起来组合散列。其他方法也可以将散列取异或在一起、用在它们之间的定界将散列连接起来、或将索引号添加到每个散列并且然后组合散列。必须知道组合散列的方法,以便使独立方复制过程并独立地得出相同的结果。

[0020] 有多种类型的TOME,包括非稀疏TOME、稀疏TOME和时钟链TOME。在非稀疏TOME中,无论数据是否在第一参考级的段(例如,秒)期间被接收到,都生成针对每个段(或时间的单位,例如一秒)的散列。因此,在非稀疏TOME中,将总会有针对每个段的散列(即,即使在六十个一秒散列中的只有三个或四个中接收到数据,六十个一秒散列也将被创建并用于创建一分钟散列)。

[0021] 在稀疏TOME中,仅当在第一参考级的段期间接收到数据时才创建散列(即,如果在特定分钟内的六十秒中的三秒内接收到数据,则只有三个散列将被散列以形成一分钟散列)。

[0022] 时钟链TOME使用创世散列。创世散列是时期中的第一个散列,并且可以是提供正被散列的数据的描述或任何其他类型的信息的文档的散列。创世散列与在第一段(例如,秒)中的数据记录进行散列。在该实施例中,第一段(例如,第一秒)的散列与在下一段(例如,第二秒)中的数据记录进行散列以创建第二段散列,第二段散列与在第三段(例如,第三秒)期间接收到的数据记录的散列进行散列等。

[0023] 散列可以在任何点处被记录在区块链中,允许将来验证数据。也就是说,通过用数据重建TOME,如果数据未损坏且未变化,则将创建相同的散列。然而,如果数据中的任何数据被改变或者数据中的时间戳已被改变,则散列将不同,表明数据中的差异。

[0024] 使用数据存储和验证平台创建的散列可以被公开地证明使用诸如公钥密码学和双向加密之类的密码技术。公钥密码学需要密钥对,其中两个密钥在数学上相链接。一个密钥是在对等网络中的节点之间被自由共享的公钥。另一个密钥是不与公众共享的私钥。公钥被用于加密明文并验证数字签名。私钥被用于解密密文并对消息进行数字签名。该消息可以由发送人的私钥进行数字签名以认证发送人的身份。然后,可以使用发送人的公钥对发送人的数字签名的交易消息进行解密,以验证发送人发起了该消息。

[0025] 数据存储和验证平台的益处包括透明性和不变性,特别是在时间敏感的数据中,因为数据存储和验证平台可以确定当数据通过相同的算法运行并且与被存储的散列进行比较时数据是否已经稍有变化。当数据的散列被记录到区块链时,数据存储和验证平台提供了不可否认性。一旦散列已经被记录在区块链中,散列数据就不会在不使散列无效的情况下被篡改。数据存储和验证平台通过允许数字签名证明被散列数据的有效性(例如公钥密码学)来提供外部证明。

[0026] 数据记录可以在任何时间间隔(例如,秒、日、星期、月、年、十年)被验证。数据存储和验证平台可以在更短的时间间隔内验证数据,从而避免需要验证可能跨越数十年的整个数据集。数据存储和验证平台可以验证任何类型的数据或任何量的数据。附加地,数据记录

所需的存储是不重要的。尽管在分布式账本上记录每笔交易是不切实际的，但是可以将某些时间点(例如，每小时、每日、每周、每月)的散列记录到由网络节点维护的分布式账本(例如，用于比特币的区块链)。

[0027]　在本公开中使用时间和具体时间增量作为示例。然而，本公开的实施例可以与任何数据一起使用并且在任何时间间隔被散列。

[0028]　此处所介绍的技术可以被具体化为专用硬件(例如电路)、用软件和/或固件适当地编程的可编程电路、或专用电路与可编程电路的组合。因此，实施例可以包括具有存储在其上的指令的机器可读介质，所述指令可以用来对计算机(或其他电子设备)进行编程从而执行过程。机器可读介质可以包括例如：软盘、光盘、光盘只读存储器(CD-ROM)、磁光盘、只读存储器(ROM)、随机存取存储器(RAM)、可擦除可编程只读存储器(EPROM)、电可擦除可编程只读存储器(EEPROM)、磁卡或光卡、闪存、或者适用于存储电子指令的其它类型的介质/机器可读介质。

[0029]　图1图示了其中可以使用本公开的一些实施例的基于网络的操作环境100的示例。如图1所图示，操作环境100包括运行在一个或多个计算设备110A-110M(诸如移动设备、移动电话、平板计算机、移动媒体设备、移动游戏设备、基于车辆的计算机、专用终端、公共终端、台式或膝上型计算机、智能手表或其他可穿戴技术、自助服务终端等)上的应用105A-105N。在一些实施例中，用于执行诸如生成文档或命令之类的操作的应用105A-105N可以被存储在计算设备上或者可以被远程地存储。这些计算设备可以包括用于通过经网络115连接到数据存储和验证平台120来接收和发送流量的机制。

[0030]　计算设备110A-110M被配置为经由网络115与数据存储和验证平台120通信。在一些实施例中，计算设备110A-110M可以从数据存储和验证平台120或数据存储125检索信息或向数据存储和验证平台120或数据存储125提交信息并且运行具有由数据存储和验证平台120检索的定制内容的一个或多个应用。例如，计算设备110A-110M各自可以执行浏览器应用或定制客户端以实现在计算设备110A-110M和数据存储和验证平台120之间的交互。

[0031]　数据存储和验证平台120可以在一个或多个服务器上运行，并且可以用于使用散列技术来创建数据记录和验证数据，将散列记录到分布式账本(ledger)，记录数字签名以及比较散列等其他活动。数据存储和验证平台120可以与数据存储125和计算设备110A-110M可通信地耦合，并且可以从计算设备110A-110M和数据存储125通信、访问或接收数据(例如，文档、交易数据)。

[0032]　数据存储和验证平台120可以基于用户需求和/或商业目标由单个公司或服务提供商定制或校准。例如，数据被散列的间隔可以基于其中数据被接收的各种时间间隔(例如，交易数据比一秒更快到达，而月度报告通常每月发生)，并且因此数据存储和验证平台120可以针对不同的用途和不同的用户进行不同的校准。

[0033]　除其他之外，数据存储和验证平台120提供创建数据记录使得数据可以在将来的任何时间被容易地验证的方法。数据存储和验证平台120可以创建TOME，其中树的叶是特定记录，并且树由每个第一分段时间段的数据记录的数量、在第一时间段中组成第二分段时间段的段的数量、在第二时间段中组成第三分段时间段的段的数量等等。例如，树可以由每秒的数据记录的数量、每分钟的秒数、每小时的分钟数、每日的小时数等来定义。数据存储和验证平台120可以在树中的任何点记录散列，并且然后将该记录与数据的散列进行比较

以验证数据没有改变。

[0034]　数据存储125可以被用于管理对诸如交易数据、文档、用户信息和其他信息之类的数据的存储和访问。数据存储125可以是使用在数据库模式中定义的类建模的一组集成对象的数据存储库。数据存储125可以进一步包括可以存储数据的平面文件。数据存储和验证平台120和/或其他服务器可以收集和/或访问来自数据存储125的数据。

[0035]　数据存储和验证平台120通过网络130与一个或多个分布式账本135可通信地耦合。

[0036]　网络115和网络130可以是相同网络或者可以是单独的网络，并且可以是使用有线和/或无线通信系统的局域网和/或广域网的任何组合。网络115或者网络130可以是或者可以使用任意一项或多项协议/技术：以太网、IEEE 802.11或WiFi、全球微波互联接入(WiMAX)、蜂窝电信(例如，3G、4G、5G)、CDMA、电缆、数字用户线(DSL)等。类似地，网络115和网络130上使用的联网协议可以包括：多协议标签交换(MPLS)、传输控制协议/互联网协议(TCP/IP)、用户数据报协议(UDP)、超文本传输协议(HTTP)、简单邮件传送协议(SMTP)和文件传送协议(FTP)。可以使用包括超文本标记语言(HTML)或可扩展标记语言(XML)的技术、语言和/或格式来表示在网络115和网络130上交换的数据。另外，可以使用常规加密技术诸如安全套接层(SSL)、传输层安全(TLS)、和互联网协议安全性(Ipsec)对所有或一些链路进行加密。

[0037]　(多个)分布式账本135或自动地(例如，在时间段的结束时)记录散列或按分布式账本上的请求记录散列。比如，比特币使用被称为区块链的分布式公共账本。当(多个)分布式账本135从数据存储和验证平台120接收到用适当密钥签名的散列并且该散列由网络节点验证时，(多个)分布式账本135将散列记录到分布式账本。

[0038]　图2图示了根据本公开的一个或多个实施例的数据存储和验证平台120内的一组组件。根据图2所示的实施例，数据存储和验证平台120可以包括存储器205、一个或多个处理器210、数据接收模块215、时间戳模块220、散列引擎225、记录模块230、数字签名模块235、数据验证模块240，比较模块245和图形用户界面(GUI)生成模块250。其他实施例可以包括这些模块和组件以及其他模块、应用、和/或组件中的一些、全部、或没有一个。仍然，一些实施例可以将这些模块和组件中的两个或更多个并入单个模块和/或可以将这些模块中的一个或多个的功能的一部分与不同模块相关联。例如，在一个实施例中，数据验证模块240和比较模块245可以被组合成单个组件。

[0039]　存储器205可以是用于存储信息的任何设备、机构、或被填充的数据结构。根据本公开的一些实施例，存储器205可以是或包括例如任何类型的易失性存储器、非易失性存储器、和动态存储器。例如，存储器205可以是随机访问存储器、存储器存储设备、光学存储器设备、磁性介质、软盘、磁带、硬盘驱动器、可擦除可编程只读存储器(EPROM)、电可擦除可编程只读存储器(EEPROM)、光盘、DVD等。根据一些实施例，存储器205可以包括一个或多个磁盘驱动器或闪存驱动器、一个或多个表格、一个或多个文件、一个或多个本地高速缓存存储器或处理器高速缓存存储器、一个或多个关系数据库或平面数据库等。另外，本领域技术人员将领会可以被用作存储器205的用于存储信息的许多附加设备和技术。

[0040]　存储器205可以被用来存储用于在(多个)处理器210上运行一个或多个应用或模块的指令。例如，在一个或多个实施例中，存储器205可以用来容纳执行数据接收模块215、

时间戳模块220、散列引擎225、记录模块230、数字签名模块235、数据验证模块240、比较模块245和GUI生成模块250的功能所需的指令中的全部或部分。

[0041]　数据接收模块215可以将数据(例如，数据项、数据记录)接收到数据存储和验证平台120中以使用散列技术来创建数据的记录。数据可以是任何类型的数据(例如，文档、视频、图片、电子邮件消息)，并且数据可以以任何间隔(例如，每秒、每小时、每年三次、随机地)被接收。在一些实施例中，由数据接收模块215接收的数据可以被保密，但仍然可被验证。例如，被发送的数据可能是提供了不透明的不可否认的方式的秘密数据的散列。在一些实施例中，由数据接收模块215接收的数据可以是API调用的结果(假设API调用的结果保持一致以用于验证第三方数据)。

[0042]　数据接收模块215还可以接收验证数据，以用于验证验证数据与初始接收到的数据相同的目的。也就是说，验证数据的记录可以使用相同的散列技术来进行以确定它是否与初始数据相同。这种验证可能有助于验证从对等网络传输的数据，以用于检测假数据或用于遵守审计。经由数据接收模块215接收的数据或数据项可以被传送到时间戳模块220以用于加时间戳(如果需要的话)，或者直接传送到散列引擎225用于散列(如果不需要时间戳的话)。

[0043]　时间戳模块220可以确定是否需要时间戳，然后根据需要对数据加时间戳。一些数据可能已包括时间戳，这可能对时间敏感的数据有帮助。在一些实施例中，不管数据是否已经包括时间戳，都对数据加时间戳。数据可以以任何粒度(例如，皮秒、毫秒、微秒、纳秒、秒、分钟、小时)被加时间戳。时间戳可以是时期的开始或数据记录的真实时间戳。被传递的时期值是时期中包含的秒数。时间可以以任何足够特定以表示时刻(全世界)(例如Julian，具有诸如GMT之类的时区的数据串)的方式来指定。具有时间戳的数据被发送到散列引擎225。

[0044]　散列引擎225接收数据，并在其时间戳记录级别处用其时间戳散列数据。然后每个记录的散列按时间顺序被组合。有时，时间分辨率的粒度不够按确定性顺序放置记录(即，两个数据记录在相同时间被接收，或者用相同的时间戳被记录)。在这种情形期间，数据记录的散列可被用于次级排序，因为在验证期间，验证数据的散列将导致与原始数据相同的散列(如果数据是相同的)，这可以被排序。因此，在一些实施例中，首先按时间对数据进行排序，然后如果存在平局(tie)，则通过数据的散列对数据进行排序。以这种方式排序允许第三方或不同的兼容系统以一致的方式排序数据的方法。

[0045]　每个参考级可以具有多个有序的段，其中的每一个被散列以创建针对参考级的单个散列，然后其被用于生成下一个参考级。作为示例，假设秒是第一参考级，分钟是第二参考级，小时是第三参考级。第一参考级包括60个有序的段(即60秒)，它们的散列将被用作第二参考级的段。随着数据被接收，数据被散列，并且在特定段(即秒)期间接收到的所有数据被散列。然后，针对每个段的散列数据(即，60个散列)以升序一起被散列以在第二参考级上生成一个段(即，60个一秒散列被散列以生成一分钟散列)。第二参考级包括60个段(即60分钟)。因此，来自第二参考级的60个有序段以升序一起被散列以在第三参考级上生成一个段(即，60个一分钟散列被散列以创建一小时散列)。第三参考级包括以升序一起被散列的24个段以在第四级上生成一个段(即，24个一小时散列被散列以创建一日散列)。这个过程可以继续，使得一个散列代表数据的数星期、数月、数季、数年、数十年或任何其他时间段。

[0046]　在每个参考级处的记录数据的每个散列段可被标记为参考级的散列段或TOME。因

此，如果第一参考级是秒，则每秒的散列是秒参考级的散列段。如果第二参考级是分钟，则被散列的秒段中的六十个的散列是分钟参考级的一个散列段。如果第三参考级是小时，则被散列的分钟段中的六十个的散列是小时参考级的一个散列段。

[0047] 当接收到数据时，数据被加时间戳并且用时间戳进行散列。然后，在第一秒内接收到的所有数据以时间排序方式被散列在一起。针对第二秒、第三秒等发生相同的过程，直到该参考级中的段的结束。

[0048] 散列引擎225可以使用稍微变化的散列算法。例如，在“稀疏TOME”中，当在任何时间段内没有接收到数据时，不会针对该时间段创建或存储散列。因此，可能针对特定的秒、分钟、小时或星期不存在散列。在稀疏TOME中，如果在没有接收到数据的时间段内的数据被要求验证时，它将返回空(null)，0或者没有任何值用于散列。例如，在一年中只具有一个数据点的稀疏TOME中，将会有针对该秒、分钟、小时、日、星期、月和年的数据，但没有其他数据(例如，sha256(sha256(sha256(sha256(sha256(sha256(one_data_point)))))))。

[0049] 散列引擎225也可以使用“非稀疏TOME”算法。在非稀疏TOME算法中，如果在任何时间点没有数据，则散列引擎225仍然创建散列。假定秒是最低参考级，如果在一秒期间没有接收到数据，则针对该秒创建散列(例如，空串的散列)。例如，可以使用空串的静态SHA256散列。在非稀疏TOME中，将总会有60个一秒散列来生成一分钟散列。

[0050] 相对非稀疏TOME使用稀疏TOME具有优势，反之亦然。例如，非稀疏TOME提供确认在该时间段内没有接收到数据的数据点，但当使用稀疏TOME时，除了缺少数据外没有特定的确认。在另一方面，因为不管数据是否被接收到，非稀疏TOME都提供散列，所以当没有在第一参考级的每个段期间都接收到数据时，非稀疏TOME的散列文件将大于稀疏TOME的散列文件。

[0051] 通常地，散列的数据存储不是大量的。例如，假设参考级包括秒、分钟、小时、星期、月和年；散列是SHA256；二进制散列被存储为32字节散列；并且被连接的散列被存储为64个十六进制字符，则一整年的非稀疏TOME数据的存储可能大约为两千兆字节的数据。取决于对于给定的秒接收和散列多少数据(例如，取决于在特定秒期间接收到的数据项的数量，针对该秒被散列的十六进制字符串可以更大或更小)，存储数量可以增加或减少。在非稀疏TOME中，分钟散列将始终是十六进制值的3840(即64*60)字符串，而星期散列是每日散列的十六进制值的448(即，64*7)字符串。

[0052] 散列引擎225还可以使用“时钟链TOME”。时钟链TOME是非稀疏TOME，因为不管对于该段，数据是否被接收，针对最低参考的每个段(例如，每秒)都有散列。时钟链TOME从非稀疏TOME中是独一无二的，因为它从第一参考级的前一段获取散列，使用来自前一段的散列作为当前段的第一个散列，并且将来自前一段的散列与针对当前段接收到的其他数据散列按升序散列。然后将当前段的散列用作随后段的第一个散列。

[0053] TOME的第一个段的第一个散列可以称为创世散列，因为它是创世元数据的散列。创世元数据可以定义与TOME数据相关的术语和/或提供对于TOME存在的推理(例如，说明被交易的资产和原始数量的法律定义、客户数量、经纪交易商的生成的文档)，而TOME数据是实际数据的散列(例如，资产的交易数据)。因此，创世散列和在第一秒期间接收到的所有数据散列被散列以形成第一秒散列，将第一秒散列与在第二秒期间接收的所有数据散列进行散列以形成第二数据散列，第二数据散列与在第三秒期间接收到的所有数据散列进行散

列，依此类推。时钟链实质上具有生日(birthday)，并且跟踪针对每秒的每个数据点，由时钟链树的生日开始。附加地，在验证期间，通过向后回溯到树的开始，时钟链TOME将验证(并且需要验证)每个散列。因此，时钟链可以验证从股票发行开始的每次交易。

[0054] 散列引擎225可以使用SHA256来创建散列，并且可以使用64个十六进制字符来表示散列。存储可以通过存储32字节的二进制散列来优化，但是要被散列的被连接散列可以是64个十六进制字符。

[0055] 记录模块230将一个或多个散列(TOME)记录到诸如区块链之类的介质中。任何区块链可以被用来记录一个或多个TOME。记录模块230可以在提交发生的任何时间间隔(例如，每分钟、通过组合分钟散列的每小时)处记录TOME。做出指定时期边界的结束(例如，秒的结束，一分钟结束)的提交并且该提交锁定数据，以便一旦被提交就不再添加更多的数据。然后记录模块230组合所有散列达到时期边界。如果提交在不是边界的时间戳上(例如，1.5秒)发生，则可以使用向后回溯的最后边界(例如1秒)。

[0056] 用于记录TOME的时间间隔可以取决于区块链的速度。例如，一些区块链可以将小时散列记录到诸如比特币之类的十分钟区块链中，或将分钟散列每分钟地记录到诸如无限币(Inifinitecoin)之类的30秒区块链中。在一些实施例中，TOME以特定间隔被自动地记录在区块链中。例如，可以记录涵盖该时期的整个数据的每小时或每日时间。因此，记录模块230记录散列以透明且永久地记录任何所选时间块的数据的存在证明。

[0057] 在一些实施例中，在通过某个时期(例如，天)记录数据之后，不可以对包括在那日(例如，每日)中的TOME的任何TOME作出改变。除区块链以外的其他介质可用于记录散列，诸如报纸、网站博客、推特等。

[0058] 数字签名模块235可以允许多个实体在任何时间级签名(即证明)该数据。证明可以每日、每月、每周和/或每年发生。为了证实数据，可以使用公钥密码学，使得ATS或经纪交易商可以数字地签名散列，公开地声明数据的散列是真实的并且代表实际数据。附加地，数字签名可以由独立地审计数据和验证散列的设备自动地添加。在一些实施例中，每个数据条目可以被数字地签名。在示例中，通过两个系统播放数据的散列，并且然后比较针对该日的散列。如果散列是相同的，则数字签名可以被应用于散列，验证针对该日的数据。在一些实施例中，散列在被记录在区块链上之前被数字签名。散列的签名表示在该时期期间发生的所有时间排序的数据都是有效的。因此，针对一星期数据的散列的签名证明整个星期的数据是有效的。

[0059] 数据验证模块240通过在一时间点接收TOME并通过相同散列算法运行潜在可疑的验证数据来验证验证数据(即，确定自从原始数据被散列后数据是否已经改变)。验证数据将包括来自与初始数据相同时间范围的数据。假设使用具有相同规格(例如，如何指定时间记录、实数的精度、数据格式(例如，JSON、XML、二进制))的相同的散列算法或函数，针对每个数据记录应该复制相同的散列，导致相同的TOME。因为时间戳被记录在记录中，所以针对秒、分钟、日、小时等的默克尔树散列也可以被复制。数据验证模块240可以验证任何类型的数据。因此，数据验证模块240可以验证并保证对于任何时期(即时间范围)数据未被添加、移除或篡改。在一些实施例中，数据验证模块240在触发事件时执行数据的验证。触发事件可以包括由请求数据完整性检查的监管者或其他实体所需的审计事件。数据验证模块240可用于将一个系统与并行运行的另一个系统进行比较。

[0060] 验证所需的数据量取决于用于创建散列的算法的类型以及用于验证的数据的时间范围。例如,为了验证使用时钟链算法散列的数据,数据验证模块240必须验证以创世元数据开始的针对验证所要求的整个时间范围。因此,用户必须提供创世元数据以及直到用户有兴趣验证的时间点之前的所有数据。对于通过设计执行的每个验证,数据验证模块240验证向后回溯到当时钟链算法开始时的树的开始的每个散列。相反,当使用非稀疏TOME或稀疏TOME时,用户可以指定要验证的数据的子集。由于对于非稀疏TOME或稀疏TOME,没有特定的创世元数据或生日,所以不需要在特定时间点开始。例如,如果用户想要验证特定小时的数据,那么整个小时的数据是必需的所有数据。数据验证模块240获取小时的数据并重新创建针对小时的散列。类似地,如果用户想要验证一年的数据,则只有该整年的数据是必需的。

[0061] 比较模块245将原始散列与由数据验证模块240创建的验证数据的散列进行比较。如果散列是不同的,则数据中的某些东西被改变。数据中的任何改变(包括时间戳的改变、数据的改变或数据的重新排序)将产生不同的散列。例如,如果数据是电影并且电影的一个像素被改变,则散列将是不同的。在另一个示例中,如果数据是交易数据并且数百万交易中的一笔交易的时间戳已被更改,则散列将是不同的。

[0062] 如果散列是不同的,则数据中的某些东西改变了。为了准确地确定数据被更改的位置,可以检查更小的时间量。例如,在非稀疏TOME或稀疏TOME中,如果一个月的数据被验证并且验证失败,则可以验证该月数据中的星期中的每个星期以确定哪个星期失败。一旦确定了失败的星期,就可以验证失败的星期中的每一天以识别哪一天失败。一旦识别了失败的天,每个小时就可以被验证以确定哪个小时失败。该过程可以迭代地继续,直到识别出失败的确切的数据散列。如果算法是时钟链,则识别已经改变的确切数据将是稍微不同的过程,因为从创世直到验证点的所有数据必须被包括。假设一个月的数据未能通过验证,可以验证从创世到该月结束的所有数据以及从创世到该月中的数据,以确定数据在上半月还是下半月期间改变。一旦确定哪一半月包含更改后的数据,则下一次迭代可以验证到该月的第一星期(如果前两星期包含不良数据)或该月的第三星期(如果后两星期包含不良数据)。可以执行如描述的迭代过程,直到识别出确切的数据记录。

[0063] 通过验证事件的顺序,在不使验证数据无效的情况下不会发生事件的重新排序。可以将比较数据提供给审计员以证明数据没有改变。因此,现代数据流的高速和高容量特性可以与按照规定所要求的每月、每季度或每年的传统的审计间隔相匹配。

[0064] GUI生成模块250可以生成允许与用户交互的一个或多个GUI屏幕。在至少一个实施例中,GUI生成模块250生成接收信息和/或向用户传达信息的图形用户界面。例如,GUI生成模块250可以显示用户界面,其中用户可以请求要被验证的数据的时间范围并且指定所使用的算法的类型。GUI生成模块250还可以显示验证结果。

[0065] 图3是图示了在数据存储和验证平台中使用的组件的交互的图300。如图3所示,数据馈送302接收有序离散数据项的高速数据。TOME 304通过使用参考级以排序的方式散列数据项来生成高速数据的TOME。记录器306在区块链上的任何时间点记录数据项的散列,使得散列的记录被记录。因此,稍后通过用数据重建被记录的散列并验证被记录的散列和被重建的散列相同,可以验证数据未被改变。证明单元308接收验证数据和/或散列准确(即,数据是未受危害的)的数字签名。证明可以稍后被用作初始数据是正确数据的进一步证明。

[0066]　图4图示了使用非稀疏TOME存储和验证数据的过程。如所示，接收数据项402、404、406、408和410，并且每个数据项被加时间戳和散列。如上所述，在非稀疏TOME中，即使在一秒期间没有接收到数据，也将创建指定在该秒期间不存在数据的散列。接下来，将每秒的数据项的散列按顺序散列在一起。例如，数据项402和404在第一秒期间被接收；因此，数据项402和404的散列被散列在一起以创建第一组的秒散列中的第一秒的散列，元素412。图4示出在第一秒期间仅发生两个数据项；然而，可以接收更少或更多的数据项，并且每个数据项将被单独地散列并随后与数据项的其他散列一起被散列。针对之后的59秒重复此过程，使得存在表示在每秒期间接收的所有数据项的60个散列。

[0067]　此后，将60个散列(每秒一个)散列在一起以创建第一分钟散列，元素416。当闰秒是可适用时，61个秒散列可被散列在一起以创建分钟散列。这个过程重复达60分钟，使得在60分钟的结束时有表示在分钟中的每个分钟期间接收到的所有数据的60个散列。如所示，数据项406、408和410在第二组秒(例如，秒61-120)期间被散列。然后数据项406、408和410的散列在元素414中被散列以生成第二组秒散列中的第一个。一旦创建了全部60秒(例如，秒61-120)的散列，则通过散列所有60个秒散列来创建第二分钟散列，元素418。

[0068]　接下来，将元素416和418(分钟数据的每个散列)与分钟数据的58个其他散列一起被散列，以生成第一小时散列，元素420(即，被散列数据的60分钟)。这个过程被重复二十三次，使得有总共24个小时散列。24个小时散列中的每个都被散列在一起以创建第一日散列(元素422)。日散列可以被创建并以任何期望的增量被散列在一起。例如，可以对365个日散列进行散列以生成整年数据的散列(元素424)。尽管在元素424中示出了365个日散列，但是可以理解天数可以是365或366，这取决于在特定年份中的天数。或者，可以散列30个日散列以生成月的散列(元素426)。尽管在元素426中示出了30个日散列，但是天数可以是28-31，这取决于在特定的月份中的天数。在一些实施例中，代替散列365个日散列，可以将十二个月散列散列在一起以创建年散列。附加地，90个日散列(或如适用于季度的91或92个日散列)可以被组合以创建季度的散列(元素428)。

[0069]　图5图示了使用稀疏TOME存储和验证数据的过程。稀疏TOME以与非稀疏TOME相同的方式工作，除了当在一时间段期间未接收到数据时不创建散列。因此，在稀疏TOME中，取决数据正在多快被接收，可能有很多缺失的散列。如所示，在第一秒(数据项502和504)和第二十五秒(数据项506、508和510)期间接收数据。对第一秒的数据散列(元素512)和对第二十五秒的数据散列(元素514)各自被散列，并且将第一秒的散列与第二十五秒的进行散列，以生成第一分钟散列(元素516)。由于在2-24或26-60秒期间没有接收到数据项，所以不创建附加的散列。相反，在非稀疏TOME中，不管数据是否在一秒期间被接收到，系统都生成散列，使得将总有要被散列的60个散列以生成分钟散列。

[0070]　然后通过散列分钟散列来创建小时散列(元素518)。在这个示例中，有可用于第一分钟(M1)和第四分钟(M4)的分钟散列，这意味着在第一分钟和第四分钟期间接收到至少一些数据。接下来，通过散列其中接收到数据的每个小时的散列来创建日散列。此处，日散列(元素520)包括第一小时的散列、第五小时的散列和第二十三小时的散列。可以通过对可用日(例如，第一日和第二日)的散列进行散列来生成该年的数据的散列，元素522。

[0071]　图6图示了使用时钟链TOME存储和验证数据的过程。时钟链TOME除了两个特征之外与非稀疏TOME相同地操作。首先，每个时钟链TOME以创世散列(元素602)开始。创世散列

可以是任何类型的数据，最好是解释TOME中包括什么数据的数据(例如，股票标识符、客户号)。创世散列和数据项604和606的散列被散列在一起以生成第一秒散列(元素614)。

[0072] 与非稀疏TOME的第二个区别在于第二秒散列以第一秒散列开始，元素608。因此，第二秒散列(元素616)是第一秒散列(元素608)和数据项610和612的散列的散列。针对每秒重复该过程。例如，第二秒散列将是第三秒散列的第一个散列。一旦完成了散列的第一个六十秒，针对六十秒(或当有闰秒时为61秒)中的每个的散列被散列以创建第一分钟散列(元素618)。然后在接下来的六十秒内接收的数据以相同的方式被散列以创建第二分钟散列。针对接下来的分钟中的每个重复该过程达总共60分钟。将六十个分钟散列进行散列以创建小时散列(元素620)。以针对元素422、424、426和428描述的相同的方式分别创建元素622、624、626和628。

[0073] 下文描述本公开的各种实施例。

1.一种计算机化方法，包括：

在第一参考级的第一参考级段期间接收数据项；

用对应的时间戳对所述数据项中的每个执行散列函数；

通过对所述第一参考级段中的每个中的所述数据项的散列根据他们各自的时间戳执行第二散列函数来生成针对所述第一参考级段中的每个的第一参考级段散列；以及

生成针对多个第二参考级段中的每个的第二参考级段散列，其中所述多个第二参考级段中的每个由预定数量的所述第一参考级段组成，其中所述多个第二参考级段散列中的每个通过对预定数量的所述第一参考级段散列执行第三散列函数来生成。

2.根据权利要求1所述的计算机化方法，进一步包括将所述第二参考级段散列中的至少一个记录到分布式账本。

3.根据权利要求1或2所述的计算机化方法，其中所述第一参考级段和所述第二参考级段是时间段。

4.根据权利要求3所述的计算机化方法，其中所述第一参考级段的时间段是秒，并且其中所述第二参考级段的时间段是分钟。

5.根据权利要求4所述的计算机化方法，其中所述第一参考级段的所述预定数量是六十或六十一。

6.根据权利要求1、2、3或4所述的计算机化方法，进一步包括针对附加参考级生成参考级段散列，其中所述附加参考级中的每个包括多个参考级段，其中所述多个参考级段中的每个包括预定数量的先前的参考级段，其中生成所述附加参考级段散列中的每个包括对所述预定数量的所述先前的参考级段中的每个的所述附加参考级段散列执行附加散列函数。

7.根据权利要求6所述的计算机化方法，其中第一接收数据项包括创世数据，其中所述第一参考级段中的每个的第一散列是除了所述第一参考级的第一段的第一散列之外的紧接先前的第一参考级段的散列，其中针对所述附加参考级段中的每个的第一散列是除了所述附加参考级中的每个的第一附加参考级段的第一散列之外的紧接先前的附加参考级段的散列。

8.根据权利要求1、2、3、4、5或6所述的计算机化方法，其中执行生成针对所述第一参考级段中的每个的第一参考级段散列，而不管在所述时间间隔期间是否接收到所述数据项中的任何数据项，其中当在第一参考级段中的一个期间没有接收到数据项时，执行占位符第

一参考级段散列。

9.一种非瞬态计算机可读存储介质,包括一组指令,所述一组指令在由一个或多个处理器执行时使机器用于:

根据时间间隔生成参考级,其中第一参考级包括预定数量的时间间隔,其中剩余参考级的时间间隔中的每个由先前的参考级的预定数量的时间间隔组成;

通过根据其中接收到数据的时间间隔对所述数据执行散列函数来创建在第一参考级处的数据的散列;

通过对在第一参考级的所述时间间隔中的每个处的数据的散列执行散列函数,直到第一参考级的所述预定数量的时间间隔,来生成第一参考级时间间隔散列;以及

通过对所述先前的参考级的所述时间间隔中的每个的散列执行散列函数直到所述预定数量的时间间隔,来生成针对所述剩余参考级的参考级时间间隔散列。

10.根据权利要求9所述的非瞬态计算机可读存储介质,其中在由所述一个或多个处理器执行时,所述一组指令进一步使所述机器将所述参考级时间间隔散列中的至少一个记录到分布式账本。

11.根据权利要求10所述的非瞬态计算机可读存储介质,其中当由所述一个或多个处理器执行时,所述一组指令进一步使所述机器用于:

处理接收到的验证数据,所述验证数据跨越所述参考级时间间隔散列中的所述至少一个的时间段;

生成针对所述验证数据的所述参考级时间间隔散列;以及

将针对所述验证数据的所述参考级时间间隔散列与所述参考级散列中的所述至少一个进行比较,以验证所述验证数据从所述数据没有变化。

12.根据权利要求9、10或11所述的非瞬态计算机可读存储介质,其中所述第一参考级的时间间隔是秒,并且其中在所述第一参考级中的时间间隔的所述预定数量是六十或六十一。

13.根据权利要求12所述的非瞬态计算机可读存储介质,其中所述剩余参考级包括第二参考级,其中所述第二参考级的时间间隔是分钟,其中所述第二参考级中的时间间隔的所述预定数量是六十。

14.根据权利要求13所述的非瞬态计算机可读存储介质,其中所述剩余参考级进一步包括第三参考级,其中所述第三参考级的时间间隔是小时,其中所述第三参考级中的时间间隔的所述预定数量是二十四。

15.根据权利要求10、11、12、13或14所述的非瞬态计算机可读存储介质,其中当由所述一个或多个处理器执行时,所述一组指令进一步使所述机器用于:

接收证明所述数据的密码签名;以及

将所述附加参考级时间间隔散列中的至少一个记录到分布式账本。

16.根据权利要求9、10、11、12、13、14或15所述的非瞬态计算机可读存储介质,其中当由所述一个或多个处理器执行时,所述一组指令进一步使所述机器用于对所述数据加时间戳,其中通过对在第一参考级的数据执行散列函数来创建数据的散列包括对具有所述时间戳的数据执行散列函数。

17.根据权利要求9、10、11、12、13、14、15或16所述的非瞬态计算机可读存储介质,其中

所述数据的第一接收数据包括创世数据，其中所述第一参考级的时间间隔中的每个的第一散列是除了所述第一参考级时间间隔的第一时间间隔的第一散列之外的紧接先前的时间间隔的散列，其中所述剩余参考级中的每个的时间间隔中的每个的第一散列是除了所述剩余参考级中的每个的第一参考级时间间隔的第一散列之外的紧接先前的时间间隔的散列。

18.根据权利要求9、10、11、12、13、14、15、16或17所述的非瞬态计算机可读存储介质，其中执行所述第一参考级时间间隔散列，而不管在所述时间间隔期间是否接收到所述数据，其中当在所述时间间隔期间没有接收到数据时，执行占位符第一参考级时间间隔散列。

19.一种数据存储和验证平台，包括：

一个或多个处理器；以及

计算机可读存储介质，具有存储在其上的指令，所述指令当由所述一个或多个处理器执行时使所述数据存储和验证平台用于：

根据时间间隔生成参考级，其中所述第一参考级包括预定数量的时间间隔，其中剩余参考级的时间间隔中的每个由先前的参考级的预定数量的时间间隔组成；

通过根据其中接收到数据的时间间隔对所述数据执行散列函数来创建在第一参考级处的数据的散列，其中所述数据以其被接收的顺序散列；

通过以时间顺序组合在时间间隔中的每个期间接收到的数据中的每个的散列，并且对在所述第一参考级的时间间隔中的每个处的数据的被组合的散列执行散列函数直到所述第一参考级的预定数量的时间间隔，来生成第一参考级时间间隔散列，

其中当在所述第一参考级的时间间隔中的任何一个时间间隔期间没有接收到数据时，生成针对第一参考级时间间隔散列的占位符散列；以及

通过以时间顺序组合所述先前的参考级的时间间隔中的每个的散列直到所述预定数量的时间间隔并且对所述先前的参考级的时间间隔中的每个的被组合的散列执行散列函数直到所述预定数量的时间间隔来生成针对所述剩余参考级的参考级时间间隔散列。

20.根据权利要求19所述的数据存储和验证平台，其中所述数据的第一接收数据包括创世数据，其中所述第一参考级的时间间隔中的每个的第一散列是除了所述第一参考级时间间隔的第一时间间隔的第一散列之外的紧接先前的时间间隔的散列，其中所述剩余参考级中的每个的时间间隔中的每个的第一散列是除了所述剩余参考级中的每个的第一参考级时间间隔的第一散列之外的紧接先前的时间间隔的散列。

21.根据权利要求19或20所述的数据存储和验证平台，其中以时间顺序组合在时间间隔中的每个期间接收到的数据中的每个的散列包括连接所述数据中的每个的散列。

计算机系统概览

[0074] 本公开的实施例包括上文已经描述的各种步骤和操作。各个这些步骤和操作可以由硬件组件执行或者可以具体化在机器可执行指令中，所述机器可执行指令可以用于使得利用所述指令编程的通用处理器或专用处理器执行所述步骤。替代地，可以通过硬件、软件、和/或固件的组合来执行步骤。这样，图7是计算机系统700的示例，本公开的实施例可以用于所述计算机系统。根据本示例，计算机系统700包括互连710、至少一个处理器720、至少一个通信端口730、主存储器740、可移除存储介质750、只读存储器760、和大容量存储设备770。

[0075] （多个）处理器720可以是任何已知的处理器。（多个）通信端口730可以是或包括例

如以下中的任意一项：用于基于调制解调器的拨号连接的RS-232端口、10/100以太网端口、或使用铜或光纤的千兆端口。可以取决于诸如局域网(LAN)、广域网(WAN)、或计算机系统700连接至的任何网络之类的网络选择(多个)通信端口730的性质。

[0076]　主存储器740可以是随机访问存储器(RAM)，或本领域中公知的任何(多个)其他动态存储设备。只读存储器760可以是用于存储静态信息(诸如处理器720的指令)的任何(多个)静态存储设备(诸如可编程只读存储器(PROM)芯片)。

[0077]　大容量存储设备770可以用来存储信息和指令。例如，可以使用硬盘(诸如Adaptec®的SCSI驱动器家族)、光盘、磁盘阵列(诸如Adaptec的RAID驱动器家族)、或任何其他大容量存储设备。

[0078]　互连710可以是或包括一个或多个总线、桥、控制器、适配器、和/或点到点连接。互连710将(多个)处理器720与其他存储器、存储设备、和通信块通信地耦合。取决于所使用的存储设备，互连710可以是基于PCI/PCI-X或SCSI的系统。

[0079]　可移除存储介质750可以是任何类型的外部硬盘驱动器、软盘驱动器、光盘只读存储器(CD-ROM)、光盘可重写(CD-RW)、数字视频盘只读存储器(DVD-ROM)。

[0080]　上文所描述的组件旨在例示一些类型的可能性。前述示例绝不应该限制本公开，因为它们仅仅是示例性实施例。

<u>术语</u>

[0081]　下面给出了贯穿本说明书所使用的术语、缩写、和短语的简洁定义。

[0082]　术语“连接”或“耦合”以及相关术语在操作性意义上使用并且不一定限于直接物理连接或耦合。因此，例如，两个设备可以直接地、或通过一个或多个中间介质或设备耦合。作为另一个示例，设备可以以可以在其之间传递信息的方式被耦合，同时彼此之间不共享任何物理连接。基于在此所提供的公开，根据前述定义本领域技术人员将领会连接或耦合存在的各种方式。

[0083]　短语“在一些实施例中”、“根据一些实施例”、“在所示的实施例中”、“在其他实施例中”、“实施例”等一般指跟随着所述短语的具体特征、结构、或特性包括在本公开的至少一个实施例中，或者可以包括在本公开的多于一个实施例中。此外，这种短语不一定指相同的实施例或不同的实施例。

[0084]　如果说明书陈述组件或特征“可以(may/can/could)”、或“可能(might)”被包括或具有特性，则这个具体组件或特性不是必须被包括或具有所述特性。

[0085]　术语“响应的(responsive)”包括完全或部分地响应的。

[0086]　术语“模块(module)”概括地指软件、硬件、或固件(或其任意组合)组件。模块典型地是可以使用(多个)指定的输入生成有用数据或其他输出的功能组件。模块可以或可以不是自含式的。应用程序(还称为“应用”)可以包括一个或多个模块，或者模块可以包括一个或多个应用程序。

[0087]　术语“网络”一般地指能够交换信息的一组互连的设备。网络可以少到局域网(LAN)上的若干私人计算机或者大到互联网(全球计算机网络)。如本文所使用的，“网络”旨在涵盖能够从一个实体向另一实体传输信息的任何网络。在一些情况下，网络可以包括多个网络，甚至多个异构网络，诸如一个或多个边界网络、话音网络、宽带网络、金融网络、服务提供商网络、互联网服务提供商(ISP)网络、和/或公共交换电话网络(PSTN)，所述网络通

过可操作以方便各网络彼此和之间的通信的网关互连。

[0088] 同样，出于图示的目的，本文在计算机程序、物理组件、和现代计算机网络内的逻辑交互的背景下描述本公开的各实施例。重要的是，虽然这些实施例描述了关于现代计算机网络和程序的本公开的各实施例，本文所描述的方法和装置同等地适用于其他系统、设备、和网络，如本领域技术人员将领会的。这样，本公开的实施例的已图示的应用不是意味着限制，而是示例。本公开的实施例适用的其他系统、设备、和网络包括例如其他类型的通信和计算机设备和系统。更具体地，实施例适用于通信系统、服务、和设备，诸如手机网络和兼容设备。此外，实施例适用于从私人计算机到大型网络主机和服务器的所有级别的计算。

[0089] 总之，本公开提供了用于存储和验证数据的新颖系统、方法、和布置。虽然上文给出了对本公开的一个或多个实施例的详细描述，在不背离本公开的精神的情况下，各种替代方案、更改、和等效例对本领域技术人员将是明显的。例如，虽然上文所描述的实施例指具体特征，本公开的范围还包括具有特征的不同组合的实施例以及不包括所描述的全部特征的实施例。因此，本公开的范围旨在涵盖落入所附权利要求书范围内的全部此类替代方案、更改、和变化，以及其等效例。因此，以上说明不应该被视为限制性的。

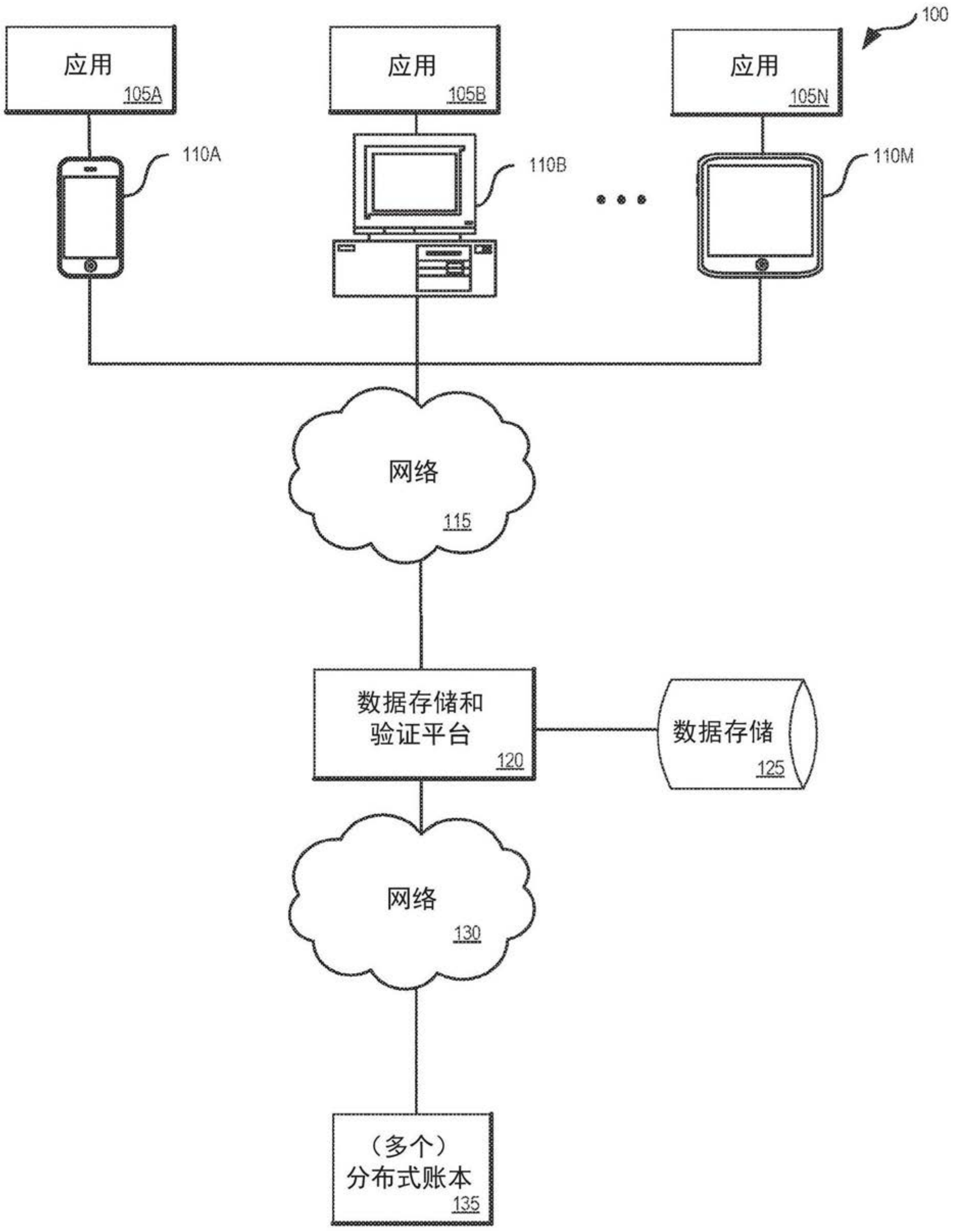

100
应用
105A
应用
105B
应用
105N
110A
110B
110M
网络
115
数据存储和
验证平台
120
数据存储
125
网络
130
（多个）
分布式账本
135

图1

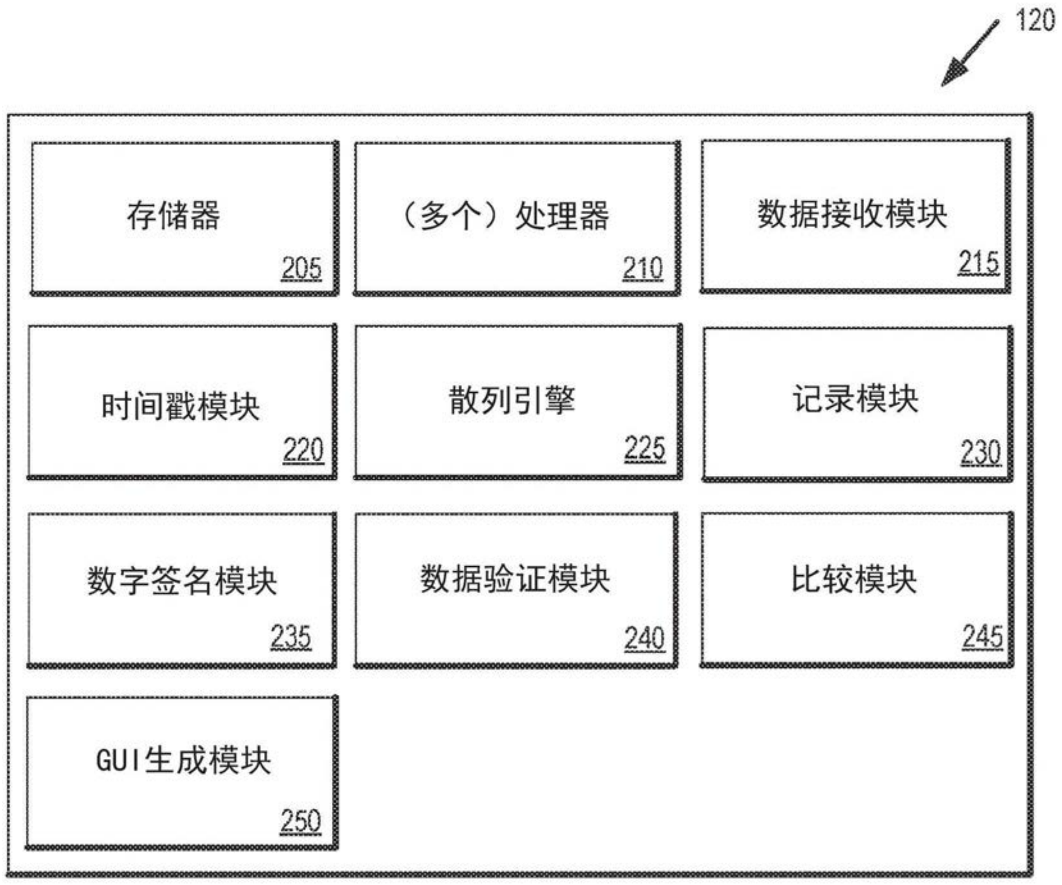

120
存储器
205
（多个）处理器
210
数据接收模块
215
时间戳模块
220
散列引擎
225
记录模块
230
数字签名模块
235
数据验证模块
240
比较模块
245
GUI生成模块
250

图2

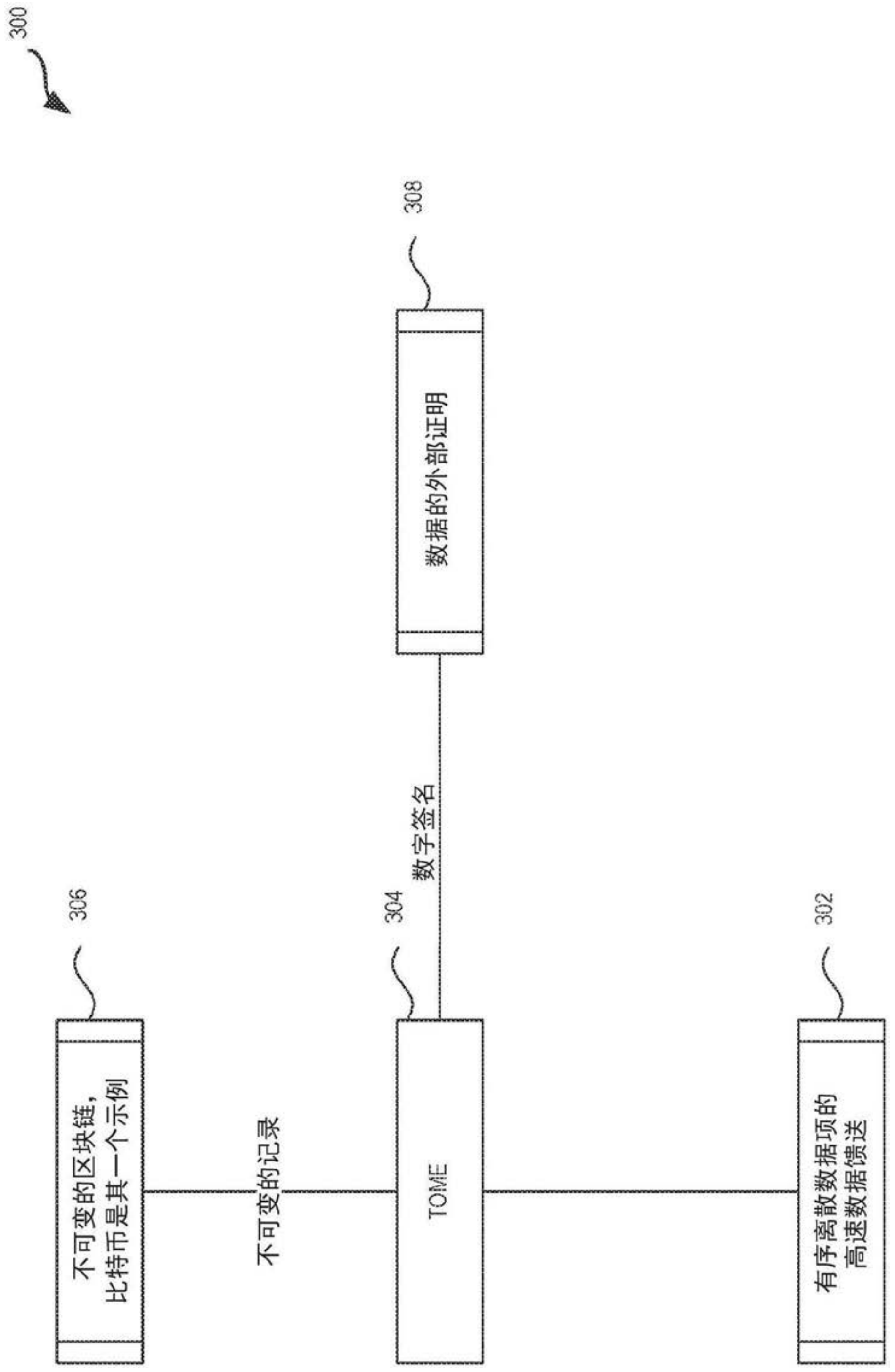
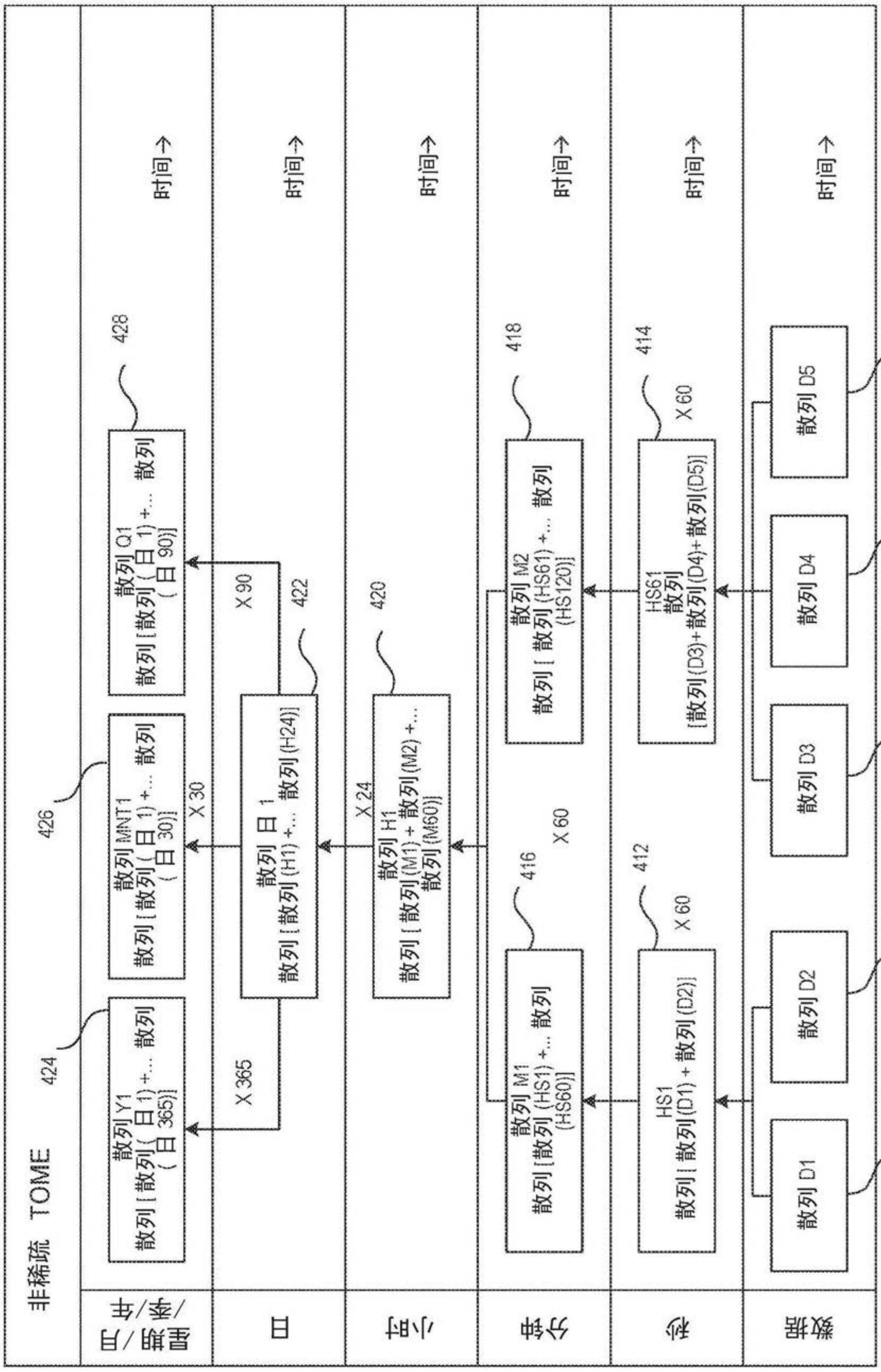

300
308
数据的外部证明
数字签名
306
304
302
不可变的区块链，比特币是其一个示例
不可变的记录
TOME
有序离散数据项的高速数据馈送

图3

说 明 书 附 图



非稀疏 TOME
星期/月/季/年
日
小时
分钟
秒
数据
时间→
散列 Y1 散列[散列(日 1)+... 散列(日 365)]
424
X 365
散列 MNT1 散列[散列(日 1)+... 散列(日 30)]
426
X 30
散列 Q1 散列[散列(日 1)+... 散列(日 90)]
428
X 90
散列 日 1 散列[散列(H1)+... 散列(H24)]
422
X 24
散列 H1 散列[散列(M1)+散列(M2)+... 散列(M60)]
420
X 60
散列 M1 散列[散列(HS1)+... 散列(HS60)]
416
散列 M2 散列[散列(HS61)+... 散列(HS120)]
418
HS1 散列[散列(D1)+散列(D2)]
412
X 60
HS61 散列 [散列(D3)+散列(D4)+散列(D5)]
414
X 60
散列 D1
402
散列 D2
404
散列 D3
406
散列 D4
408
散列 D5
410

图4

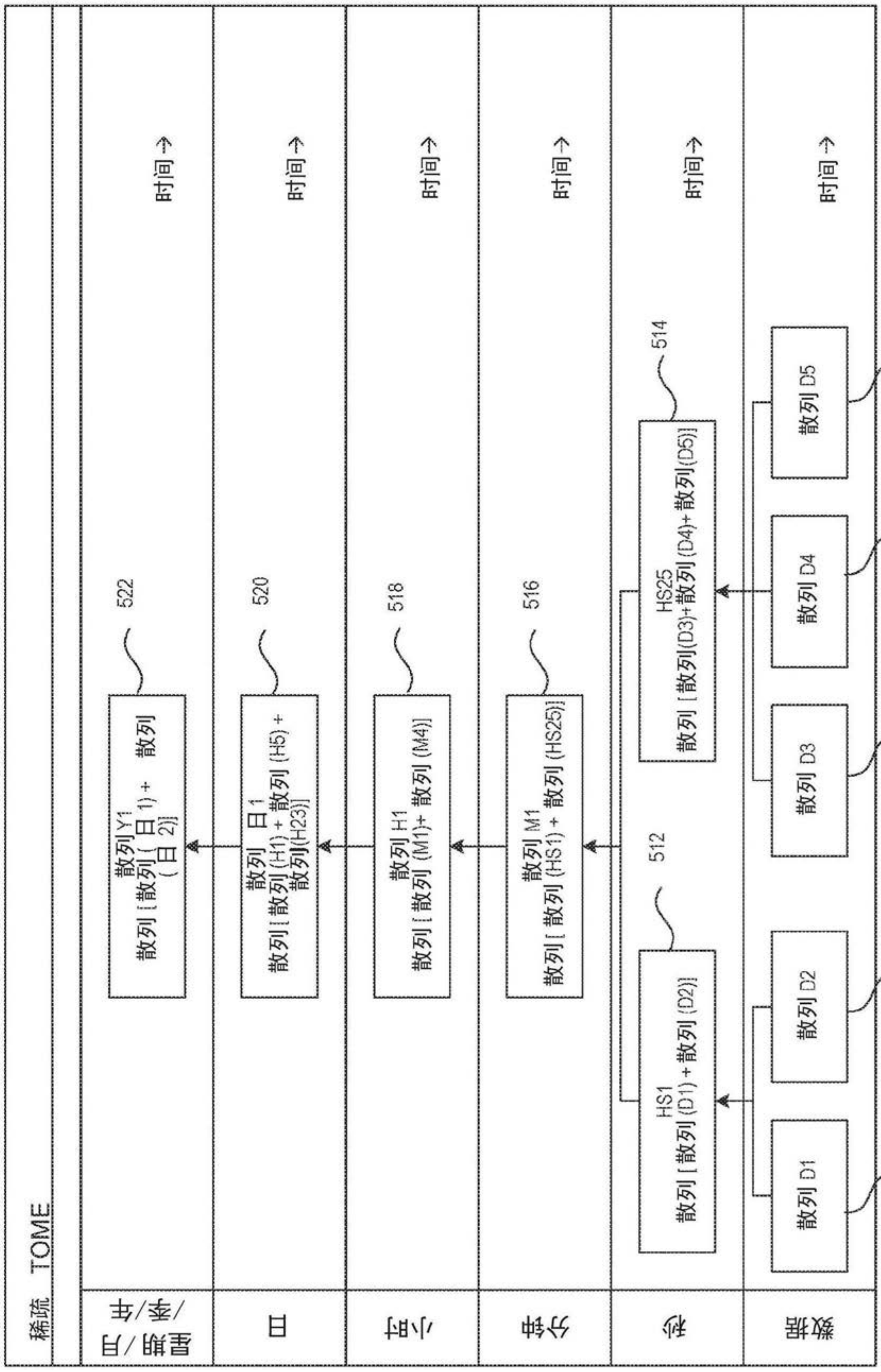

稀疏
TOME
星期/月/季/年
日
小时
分钟
秒
数据
时间→
散列Y1
散列[散列(日1)+ 散列(日2)]
522
散列 日1
散列[散列(H1)+散列(H5)+散列(H23)]
520
散列H1
散列[散列(M1)+散列(M4)]
518
散列M1
散列[散列(HS1)+散列(HS25)]
516
HS1
散列[散列(D1)+散列(D2)]
512
HS25
散列[散列(D3)+散列(D4)+散列(D5)]
514
散列D1
502
散列D2
504
散列D3
506
散列D4
508
散列D5
510

图5

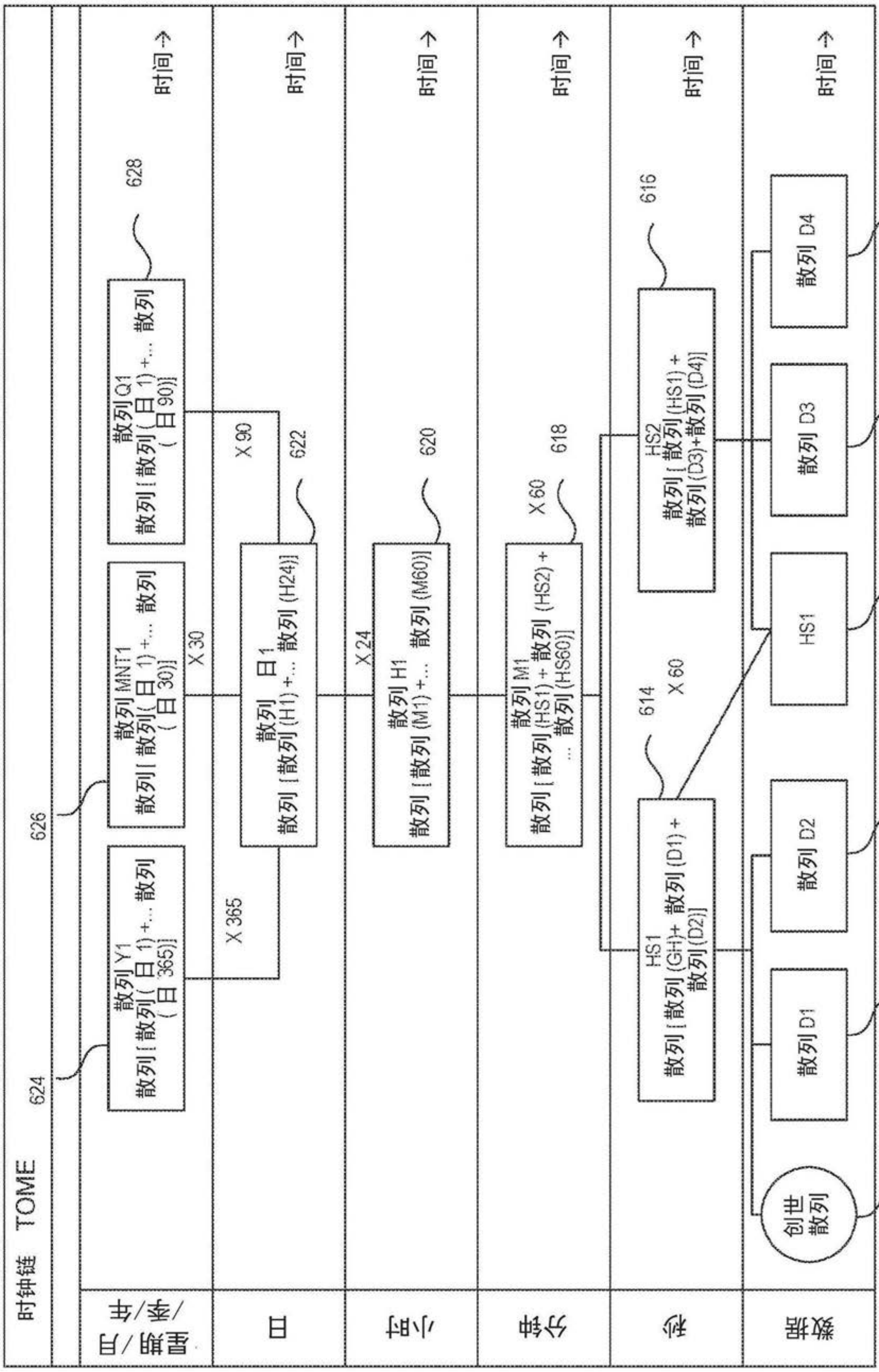

时钟链 TOME
星期/月/季/年
日
小时
分钟
秒
数据
时间→
时间→
时间→
时间→
时间→
时间→
创世散列
602
散列 D1
604
散列 D2
606
HS1
608
散列 D3
610
散列 D4
612
HS1 散列[散列(GH)+散列(D1)+散列(D2)]
614
X 60
HS2 散列[散列(HS1)+散列(D3)+散列(D4)]
616
散列 M1 散列[散列(HS1)+散列(HS2)+ ... 散列(HS60)]
618
X 60
散列 H1 散列[散列(M1)+... 散列(M60)]
620
X 24
散列 日1 散列[散列(H1)+... 散列(H24)]
622
X 365
X 30
X 90
散列 Y1 散列[散列(日1)+... 散列(日365)]
624
散列 MNT1 散列[散列(日1)+... 散列(日30)]
626
散列 Q1 散列[散列(日1)+... 散列(日90)]
628

图6

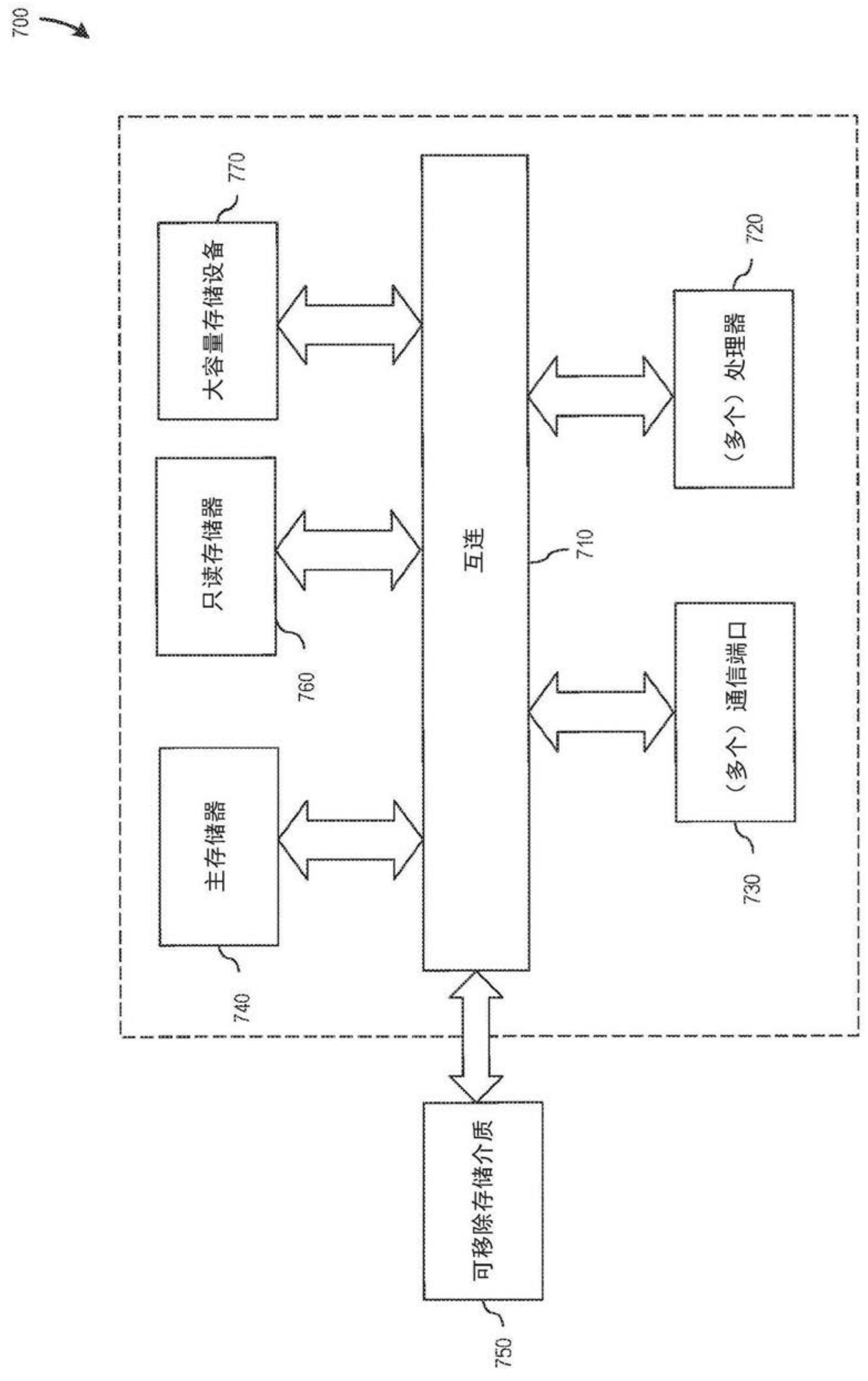

700
大容量存储设备
770
只读存储器
760
主存储器
740
互连
710
（多个）处理器
720
（多个）通信端口
730
可移除存储介质
750

图7