(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 3 790 224 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
10.03.2021 Bulletin 2021/10

(51) Int Cl.:
H04L 9/32 *(2006.01)*

(21) Application number: **19020509.6**

(22) Date of filing: **04.09.2019**

(84) Designated Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR
Designated Extension States:
BA ME
Designated Validation States:
KH MA MD TN

(71) Applicant: **I25S ApS**
3400 Hillerød (DK)

(72) Inventor: **The designation of the inventor has not yet been filed**

(74) Representative: **Tzeuton Tchangoum, Eric Olivier**
SOTERYAH IP
199 rue Hélène Boucher
34170 Castelnau-Le-Lez (FR)

(54) SPARSED MERKLE TREE METHOD AND SYSTEM FOR PROCESSING SETS OF DATA FOR STORING AND KEEPING TRACK OF THE SAME IN A SPECIFIC NETWORK

(57) The invention relates to a computer-implemented method for processing sets of data for storing and keeping track of the same in a specific network, the method implementing a Sparsed Merkle tree. The method comprises steps for

- dividing at least one of the hash values into multiple sections of rim-keys (L, A) of a given bit-width; and
- determining leave rim-keys (L) being archive leaves of the Sparsed Merkle tree, and parent rim keys (A) supporting the creation of a subtree (SMT).

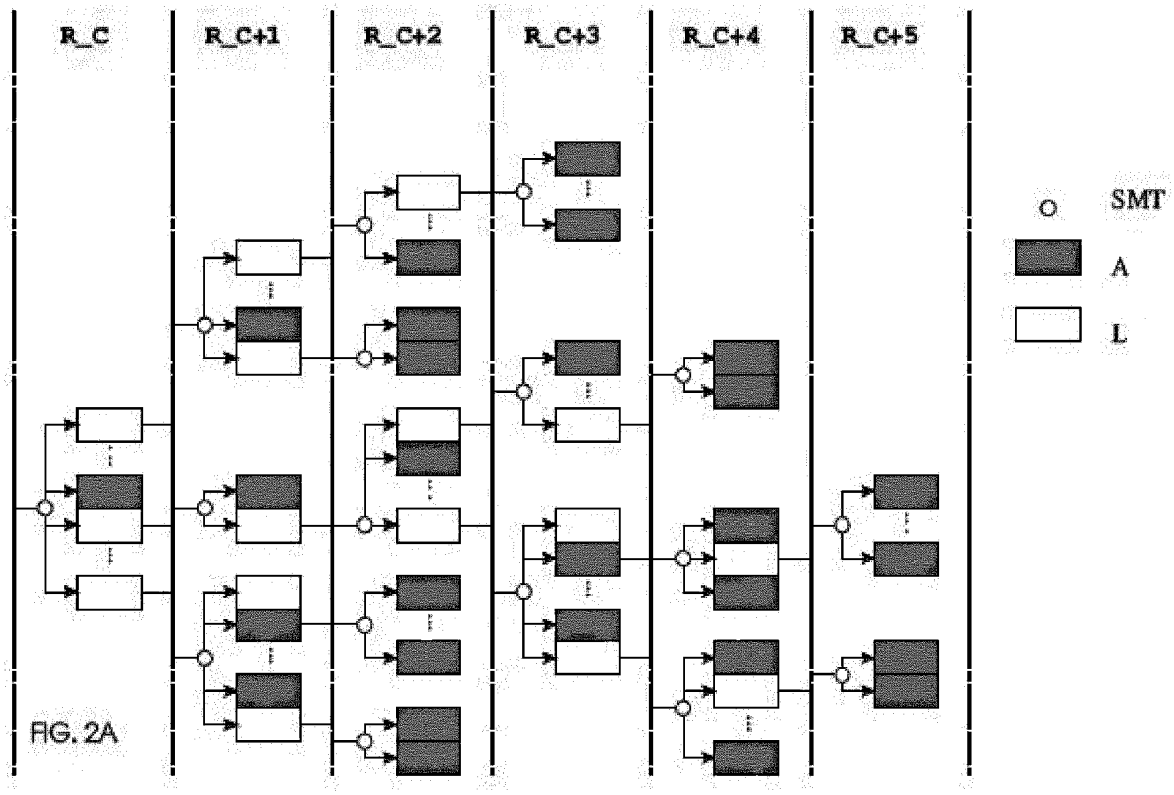


FIG. 2A

EP 3 790 224 A1

Description

[0001] The present invention relates to the field of data-storage management in databases, in particular for databases of transactions. The transactions may be relating to cryptocurrencies, such as blockchain transactions. The invention relates more particularly to storing and keeping track of transactions in a specific network.

[0002] The number of transactions is high and the transactions should be highly secured. This raises an issue of the safe and optimal storage of the data.

[0003] One solution provided in the prior art is using Merkle trees. The Merkle trees enable to cryptographically commit to a set of data. Each piece of data in the set is first hashed, and then the hashing is done again to all the data and all the way up the tree until a root node is obtained. In the illustration of figure 1, each data such as a transaction (T_A, T_S, T_X, T_Z), is hashed to obtain a hash of the transaction ($H(T_A)$, $H(T_S)$, $H(T_X)$, $H(T_Z)$). Then, consecutive hashes (($H(TA)$, $H(T_S)$) are hashed together to obtain another hash value ($H\ (H(T_A) \parallel H(T_S))$..., and so on and so forth until reaching the root node ($H(H((H(T_A) \parallel H(T_S)) \parallel H(H(T_X) \parallel H(T_Z)))))$.

[0004] The root of this tree is a hash. It can be used to show what has been called the Merkle proof, i.e. prove that some content is part of the tree. This is generally called the Proving Inclusion. For example, in order to prove that T_A is part of the tree, the siblings of T_A may be used to recompute all the tree, and check if the items match. Indeed, with $H(T_S)$ and $H\ (H(T_X) \parallel H(T_Z))$, it is easy to recomputed the original root hash and prove that T_A is part of the tree. This item may include information on a corresponding transaction to securely and efficiently store and/or retrieve the same. Using a Merkel tree saves bandwidth and processing activity in microcontrollers and the like, as it is not needed to store and re-provide the entire tree or the entire set of data.

[0005] Ideed, the Merkle tree reduces the information needed to send between two units to verify whether data is the same. Instead of exchaning and verifing each byte of information between two computer units you need to calucated the Merkle tree on a collection of data and then travers the Mekle tree and compare the hash values in the Merkle tree.

[0006] The Merkle tree is however lacking the possibility to prove that an item is not part of the tree. This is generally called the Proving Non-Inclusion.

[0007] One solution provided in the prior art is using Sparsed Merkle trees. The Sparsed Merkle tree is similar to the Merkle tree, but the contained data is indexed, and each data point is placed at the leaf corresponding to the datapoint's index. In the example of figure 1, every datapoint would have for example an ordinate value, such as (A, B, C, D) in place of (TA, T_S, T_X, T_Z) respectively, in figure 1. In order to prove that T_C is part of the tree or not, the additional check would be on the ordinate index in comparison to T_D for example.

[0008] An objective of the present invention is to provide additional means for improving the efficiency of the processing of the information on the transactions and, in particular, its storage in a specific database. More specifically, an objective of the present invention is to efficiently handle removal and adding of transactions in a secure and distribute manner. This is *interallia* because of the way it is structured it also sorts/orders the hash keys. This makes it fast to search each transaction archive.

[0009] In order to meet this objective, the invention relates to a computer-implemented method for processing sets of data for storing and keeping track of the same in a specific network, the method implementing a Sparsed Merkle tree, the method comprising steps for

- cryptographically hashing the data to obtain hash values of the data;
- where applicable, cryptographically hashing consecutive hash values of the data to obtain combined hash values of a first stage;
- where applicable, cryptographically hashing combined hash values of a first stage to obtain combined hash values of a second stage;
- until obtaining a root hash value.

[0010] According to a first aspect, the computer-implemented method is characterized by steps for

- dividing at least one of the hash values into multiple sections of rim-keys of a given bit-width; and
- determining leave rim-keys being archive leaves of the Sparsed Merkle tree, and parent rim keys supporting the creation of a subtree.

[0011] Advantageously, the method enables to limit the number of subtree contrary to the methods of the prior art making subtrees on all the hash values as long as a root hash value is not obtained. This leads to less processing calculation and less memory used in the processing of the data, because each subtree is a hash calculation.

[0012] Moreover, as the hash key of an archive is a hash of the archive and we can do a fast tree search via the rim-keys.

[0013] According to other aspects of the method taken individually or combined in any technically possible combination:

- the method comprises creating a subtree on a parent rim-key if an archived data has the same rim-key as the parent rim-key and/or;
- the method comprises dividing multiple hash values, preferably every hash values, into multiple sections of rim-keys of the given bit-width and/or;
- at least one subtree is a vector of hash of a given index and/or;
- the index is the respective rim key and/or;
- the Sparsed Merkle Tree comprises vectors of hashes and/or;
- the data are transactions.

[0014] The invention further relates to a system for processing sets of data for storing and keeping track of the same in a specific network, the system comprising means for carrying out the steps of the method according to the invention.
[0015] More generally, the invention further relates to a system for processing sets of data for storing and keeping track of the same in a specific network, the system implementing a Sparsed Merkle tree, the system comprising

- means for cryptographically hashing the data to obtain hash values of the data;
- means for cryptographically hashing consecutive hash values of the data to obtain combined hash values of a first stage;
- means for cryptographically hashing combined hash values of a first stage to obtain combined hash values of a second stage;
- means for obtaining a root hash value.

[0016] According to a first aspect, the system is characterized by

- means for dividing at least one hash value into multiple sections of rim-keys of a given bit-width; and
- means for determining leave rim-keys being archive leaves of the Sparsed Merkle tree, and parent rim keys supporting the creation of a subtree.

[0017] The advantages mentioned above for the method also apply to the system implementing the method.
[0018] According to other aspects of the system taken individually or combined in any technically possible combination:

- the system comprises means for creating a subtree on a parent rim-key if an archived data has the same rim key as the parent rim-key and/or;
- the system comprises means for dividing multiple hash values, preferably every hash values, into multiple sections of rim-keys of the given bit-width and/or;
- at least one subtree is a vector of hash of a given index and/or;
- the index is the respective rim key and/or;
- the Sparsed Merkle Tree comprises vectors of hashes.

[0019] The invention also concerns a database comprising means for storing and keeping track, in a specific network, of sets of data processed through a method according to the invention, or by means of a system according to the invention.
[0020] More generally, the invention also concerns a database for storing and keeping track of sets of data in a specific network, implementing a Sparsed Merkle tree, comprising

- hash values of the data obtained by cryptographically hashing the data;
- combined hash values of a first stage obtained by cryptographically hashing some of the consecutive hash values of the data;
- combined hash values of a second stage obtained by cryptographically hashing some of the combined hash values of a first stage;
- a root hash value

characterized by

- rims-keys of a given bit-width obtained by dividing at least one hash value into multiple sections of rim-keys; and among which are leave rim-keys being archive leaves of the Sparsed Merkle Tree, and parent rim keys supporting the creation of a subtree.

[0021] The advantages mentioned above for the method and the system also apply to the database including the corresponding data.

[0022] Another object of the invention is a network comprising one or more central unit, in particular one or more computerized central unit, and connection to additional command units implementing transactions, in particular computerized command units, the central unit(s) comprising a system according to the invention and a database according to the invention.

[0023] The invention will now be presented in details via the description of non-limitative embodiments of the invention, and based on the enclosed drawings, among which:

- figure 1 is a schematic illustration of a Sparsed Merkle Tree in a method according to the prior art;
- figure 2A is a schematic illustration of an example of a Section hierarchical Sparsed Merkle Trees in a method according to an embodiment of the invention;
- figure 2B is a schematic illustration of a Sparsed Merkle Trees vector of the method illustrated in figure 2A;
- figure 2C is an illustration of a system according to an embodiment of the invention;
- figure 3A is a schematic illustration of a list of transactions with hash values;
- figure 3B is a schematic illustration of an example of how the transactions of figure 3A are archived in the hierarchical sub-trees; and
- figure 4 is a graphical illustration of the database according to an embodiment of the invention.

[0024] The invention relates to management and storage of databases, in particular for databases of transactions, such as those relating to cryptocurrencies. The invention finds applications in blockchain transactions and the like.

[0025] The invention relates more particularly to storing and keeping track of transactions in a specific network.

[0026] The invention aims first at managing high numbers of transactions in a secure way, in order to have a safe and optimal storage of the data.

[0027] The invention further aims at overcoming the lacks of the solutions of the prior art, and in particular at improving the efficiency of the processing of the information on the transactions and, more particularly, its storage in a specific database.

[0028] More specifically, the invention aims at efficiently removing and adding transactions in a secure and distribute manner. More generally the invention relates to a CRUD (create, read, update, delete) DHT (Distribute Hash Table) system.

[0029] The invention concerns a computer-implemented method for processing sets of data for storing and keeping track of the same in a specific network. The method is based on implementing a Sparsed Merkle Tree. The Sparsed Merkle Tree as such is known to the person skilled in the art. Thus, the description will focus on the innovative features and not much on the detailed features of the prior art.

[0030] The invention is implemented in a computer based system 1a that enables a Distribute Archive of Random Transactions. Thus, the system may be called the DART system, and the method, the DART method.

[0031] The method of the invention comprises a step for cryptographically hashing the data to obtain hash values of the data. The hash values obtained are then also hashed, and so on until a root hash value is obtained. It is preferably consecutive hash values of the Sparsed Merkle tree that are hashed. The hash value obtained may be labeled as combined hashed value of the first stage. Then, a similar step is implemented to obtain combined hash values of the second stage, and so on, until the root hash value is obtained.

[0032] The data are in particular relating to transactions. More particularly, each transaction is stored in the DART system as a distribute hash-table using a crypto-graphic hash of the transaction Tdata.This means that each transaction is identified by a unique hash value h. The transaction is put into a table order via the numerical value of the hash.

[0033] In a preferred embodiment, the h value may be obtained by the formula:

$$h = H(T), h \quad [0 : 2^{N-1} - 1], N \quad N$$

Equation 1

wherein

H is cryptography hash function,
N represents the number of bits of the hash value h.

[0034] Then

$$h=\sum_{i=0}^{N-1} b_i 2^i, b_i \quad \{0,1\}$$

Equation 2

wherein the b_i sad is the *i'th* bit of the hash value h.

[0035] According to a first aspect, the method of the invention comprises a step for dividing at least one of the hash values into multiple sections of rim-keys (*L, A)* of a given bit-width. The method further comprises a step for determining leave rim-keys *L* being archive leaves of the Sparsed Merkle tree, and parent rim keys *A* supporting the creation of a subtree (*SMT*).

[0036] Advantageously, the method enables to limit the number of subtrees that are made on the Sparsed Merkle tree. Indeed, contrary to the methods of the prior art making subtrees on all the hash values as long as a root hash value is not obtained, in the invention only some rim keys become leaves and others are bases of subtrees. This leads to less processing calculation and less memory used in the processing of the data.

[0037] In particular, the hash-table is divided *m* into sections *S*. Each section contains an ordered list of hash-values within a limited range as in the following formula of a preferred embodiment, such that the sections are buildup of a hierarchy of Merkle-Trees divided in what is called rims.

$$S=\sum_{i=0}^{M-1} b_i 2^i, S \quad [0:2^M-1], m=2^M$$

Equation 3

[0038] The method of the invention further comprises a step for determining leave rim-keys being archive leaves of the Sparsed Merkle tree, and parent rim keys supporting the creation of a subtree.

[0039] In particular, the bit-width of a rim is for example *K*. Each leave in the tree is expressed via a rim-index called the rim-key. The maximum number of branches is defined as k, and thus:

$$k=2^K$$

Equation 4

[0040] If the sections bit-width *M* is selected as cardinal multiple *C* number of *K,* the rim key can be expressed as:

$$R_n=\sum_{i=0}^{K-1} b_{nK+i} 2^i$$

Equation 5

[0041] The section can be expressed as the first *C* rim-keys.

$$S=\sum_{j=0}^{C-1} R_j$$

Equation 6

[0042] From this the hash value h can be composed as:

$$h = S + \sum_{j=C}^{G-1} R_j = \sum_{j=0}^{G-1} R_j, N = KG$$

Equation 7

[0043] Figures 2A and 2B may be used to illustrate an embodiment of the invention. They show an example of a Section hierarchical Sparsed Merkle Trees. The rims are labled R_C for the rim C, R_C+1 for the following rim...

[0044] When a transaction is stored in the DART system the hash value h is calculated and the hash value is divided into sections of rim-keys (*L, A*).

[0045] According to a preferred embodiment the Sparsed Merkle Tree comprises vectors of hashes. More particularly, each sub-tree is represented by vector table the index into the vector is the rim-key of the respective rim. Vectors of hashes are very efficient means in the implementation of the invention.

[0046] Indeed, the subtree is a small hash of the vector that is fast to calculate. And even faster of the vector is spares (most null value). The deeper we go in the tree the more sparsed the subtree vectors are. With the current implementation we are working in rim 3, and rim 3 only have few none null elements with more than 100^6 archives.

[0047] According to an embodiment, the method comprises creating a subtree (*SMT*) on a parent rim-key A if an archived data has the same rim-key as the parent rim-key A.

[0048] More particularly, if an existing archived transaction has the same rim-key a sub-tree must be created.

[0049] If the rim-key is different from an already archive transaction the archive is stored in the vector-table with the current rim-key. This is an efficient way to limit the number of subtrees.

[0050] Moreover, the hash can be calculated from the archived data. In our implementation a vector element can either contain

- a hash of the including subtree branch; or
- the archived data.

[0051] It is preferable to have the minimum number of nodes to store in one complete sector.

[0052] In our implementation, the vector also includes an index pointer which points to a record on the disk. The Sector rims are stored in memory and all the sectors sub-trees are store on the local disk space.

[0053] In figure 2A, a hierarchical tree structure is shown. The white boxes *A* are hash values of the Sparsed Merkle Tree. The hatched boxes are archive leaves *L*. The circles are subtrees of the Spased Merkle Tree. Figure 2B illustrates the SMT vector.

[0054] An example illustrated in figure 3A and 3B, shows how a list of transaction is archived into the tree structure.

[0055] Because most modern computes are based on multiple 8-bits (bytes) *k*=8 in this example and *C*=2.

[0056] Supposing a list of transactions is given with the hash values listed in figure 3A. In the figure, the arrows point to a branch or an Archive. If the rim split is a branch if not it's an archive. An archive is a Leave. In the drawing a brick-dashed symbol means a branch hash. If there is none it means that in an archive the hash is calculate from data in the archive.

[0057] Figure 3B shows where the transaction archives are placed in the tree structure.

[0058] In figure 3B, the hatched squares are hash values of the SMT of a sub-branch; the other squares are Brance rim keys. The sub-tree vector element can contain a Brance (which points to a subtree) or an archive which contains the data of the archive. The rims are labeled R_0 for the first rim, R_1 for the following rim...

[0059] As it is noted before the trees are divided into sections which are defined as the first C rims. In the section rims no transaction leaves are allowed. Transactions must be archived in rim greater than C. This rule makes it easier to split the database up into a distributed database.

[0060] The complete database can be illustrated as a circular structure as shown in figure 4. Each black rectangle represents the structure shown in Figure 2A and all the center of the data shows the Merkle tree for the whole data base.

[0061] The database can be distributed between computer nodes CN in the flowing manner as a large hashtable where the hash-value h is key index key to the hash-table.

[0062] The hash-table is distributed between the computer nodes connected via a network, where each node manages a sample of sections called a DART angle.

[0063] Because the numerical value of the hash's is ordered around the circle.

[0064] For example, the maximum value of 256 hashes is 2^{256}-1.

[0065] The number rolls around from 2^{256}-1 to 0. This mean it can be plotted on a circle.

[0066] A section is to be managed by more than Q nodes to keep redundancy and security of the data.

[0067] Each node must maintain the database sections within nodes section angle, this means adding and removing the transaction and update the Merkle-Tree root of the section hash the Merkle root of the DART is named bullseye.

[0068] An example of the DART is shown in Figure 4 illustrated. The archives (transactions) are stored in a hierarchical tree structure. The archives are stored as the leaves of the tree. CN refer to the node with their number. "sSMT" refer to sector Merkle trees. "cSMT" refer to central Merkle trees. "sH" refer to Sector Hash. "be" refer to a bullseye.

[0069] Regarding now, the Sparsed Merkle Tree itself, due to the structure of the SMT the Merkle root of the complete DART can be calculate with very few hash operations compared to full Merkle tree.

[0070] In the flowing the method is describe how to calculate the Merkle tree in an efficient manner.

[0071] The function below defines hash function of two value pairs.

$$(y_L, y_R) = \begin{cases} 0 for\ y_L = 0 \wedge\ y_R = 0 \\ y_L for\ y_R = 0 \wedge\ y_L \neq 0 \\ y_R for\ y_L = 0 \wedge\ y_R \neq 0 \\ H(y_L\ \ y_L) for others \end{cases}$$

Equation 8

[0072] This definition is important because this is what makes the calculation of the hash very fast on highly sparsed vector because most of the elements is null and on null element you don't do any calculations. It also illustrated in the examples below.

[0073] The Sparsed Merkle Tree is can be represented as a vector of hashes as previously described where some of the value in the vector often is zero.

$$V_{(m,n)}|_{n<n} = [h_m, h_{m+1}, ..., h_{n-1}]$$

Equation 9

[0074] Hash function of the vector can be defined as:

$$(V_{(0,k)}) = \begin{cases} (\ (V_{(0,k/2)}),\ \ (V_{(k/2+1,k)})) & for\ k>2 \\ (h_0, h_1) & others \end{cases}$$

Equation 10

[0075] The function is of tree vector is defined as the Sparsed Merkle root hash value.

[0076] Examples of some calculations.

Example 1.

[0077] The value of a null vector will result in the value 0 which also proves that the SMT is empty.

Example 2.

[0078] If the vector *V(0,k)* only have one value *hi* which none zero then the value will be h_i.

$$(V_{(0,k)})^2|_{[0..k-1]} = V_{(0,k)}\ \ V_{(0,k)} = h_i^2 \cdot h_j|_{j=i} \neq 0\ \ \ h_j|_{j \neq i} = 0\ \ \ j\ \ \ [0..k-1]$$

Équation 11

Example 3

[0079] If the vector $V_{(0,k)}$ only contains two and only two none zero value h_i and h_j the can be expressed as:

$$\begin{aligned}\left(V_{(0,k)}\right)\Big|_{<j\ \ j\ \ [0..k-1]} &= \left(h_i, h_j\right)\\ &= H\left(h_i\ \ h_j\right),\quad h_l\Big|_{=i\ \ l=j}\neq 0\quad h_l\Big|_{\neq i\ \ l\neq j}=0\quad l\quad [0..k-1]\end{aligned}$$

Equation 12

Example 4

[0080]

$$\begin{aligned}\left(V_{(0,8)}\right) &= \left(\left[h_0, h_1, 0, 0, h_4, 0, h_6, 0\right]\right)\\ &= \left(\ \left(\left[h_0, h_1, 0, 0\right]\right),\ \left(\left[h_4, 0, h_6, 0\right]\right)\right)\\ &= \left(\ \left(\ \left(\left[h_0, h_1\right]\right),\ \left([0,0]\right)\right),\ \left(\ \left(\left[h_4, 0\right]\right),\ \left(\left[h_6, 0\right]\right)\right)\right)\\ &= \left(\ \left(\ \left(h_0, h_1\right), 0\right),\ \left(\ \left(h_4, 0\right),\ \left(h_6, 0\right)\right)\right)\\ &= \left(\ \left(h_0, h_1\right),\ \left(h_4, h_6\right)\right)\end{aligned}$$

Equation 13

[0081] This results in the following hash value

$$h = H\left(T\right), h\quad \left[0:2^{N-1}-1\right], N\quad N$$

Equation 14

[0082] As an example, to compare normal Merkle tree calculations to the SMT method and example of 4 10^9 2^{32} archives stored in the database. This will would in worst case amount to 4 10^9 hash operations.

[0083] With a good probability and archive would be store at rim 4 and an archive stored a rim 4 would about to around 4 2561000.

[0084] The invention further concerns a system 1a for implementing the method as described above. The system 1a is a computerized system and comprises inherent hardware and software modules configured to implement the method described above.. This may be called a DART system 1a. The system enables to process sets of data for storing and keeping track of the same in a specific network, the system comprising means for carrying out the steps of the method according to the invention.

[0085] The system comprises means for cryptographically hashing the data up to the obtention of a root hash value. These means include one or more microcontrollers, hardware and software modules that are known as such to the skilled person, but are however configured to implement the specific Sparsed Merkle tree of the invention.

[0086] According to a first aspect, the system comprises

- - means for dividing at least one hash value into multiple sections of rim-keys of a given bit-width; and
- - means for determining leave rim-keys being archive leaves of the Sparsed Merkle tree, and parent rim keys supporting the creation of a subtree.

[0087] The system is configured to execute the corresponding operation(s) through the microcontroller and software and hardware modules.

[0088] The advantages mentioned above for the method also apply to the system implementing the method.

[0089] The invention relates also to a database 2a comprising means for storing and keeping track, in a specific

network, of sets of data processed through a method as described above, or by means of a system as described above. This may be called a DART database 2a. The database include one or more microcontrollers, hardware and software modules that are known as such to the skilled person, but are however configured to implement the invention.

[0090] More generally, the database comprises hash values of all the data of one or more Sparsed Merkle tree(s) all the way to the corresponding root hash value(s).

[0091] In addition to that, the database comprises

- rims-keys of a given bit-width obtained by dividing at least one hash value into multiple sections of rim-keys; and
- among which are leave rim-keys being archive leaves of the Sparsed Merkle tree, and parent rim keys supporting the creation of a subtree.

[0092] The advantages mentioned above for the method and the system 1a also apply to the database including the corresponding data.

[0093] Another object of the invention is a network comprising one or more central unit(s) 3a, in particular one or more computerized central unit, and connection(s) 4a to additional command units 5a implementing transactions, in particular computerized command units, the central unit(s) 3a comprising a system 1a and a database 2a as described above. This may be called a DART network. The network may be connected via the internet to the additional command units.

Claims

1. Computer-implemented method for processing sets of data for storing and keeping track of the same in a specific network, the method implementing a Sparsed Merkle tree, the method comprising steps for

   - cryptographically hashing the data to obtain hash values of the data;
   - where applicable, cryptographically hashing consecutive hash values of the data to obtain combined hash values of a first stage;
   - where applicable, cryptographically hashing combined hash values of a first stage to obtain combined hash values of a second stage;
   - until obtaining a root hash value (be),

   characterized by

   - dividing at least one of the hash values into multiple sections of rim-keys (L, A) of a given bit-width; and
   - determining leave rim-keys (L) being archive leaves of the Sparsed Merkle tree, and parent rim keys (A) supporting the creation of a subtree (SMT).

2. Computer-implemented method according to the preceding claim, comprising creating a subtree on a parent rim-key (A) if an archived data has the same rim-key as the parent rim-key.

3. Computer-implemented method according to any of the preceding claims, **characterized by** dividing multiple hash values, preferably every hash values, into multiple sections of rim-keys(L, A) of the given bit-width.

4. Computer-implemented method according to any of the preceding claims, wherein at least one subtree (SMT) is a vector of hash of a given index.

5. Computer-implemented method according to the preceding claim, wherein the index is the respective rim key.

6. Computer-implemented method according to the preceding claim, wherein the Sparsed Merkle Tree comprises vectors of hashes.

7. Computer-implemented method according to any of the preceding claims, wherein the data are transactions.

8. System (1a) for processing sets of data for storing and keeping track of the same in a specific network, the system implementing a Sparsed Merkle tree, the system comprising - means for cryptographically hashing the data to obtain hash values of the data; - means for cryptographically hashing consecutive hash values of the data to obtain combined hash values of a first stage; - means for cryptographically hashing combined hash values of a first stage to obtain combined hash values of a second stage; ...; - means for obtaining a root hash value, **characterized by** - means

for dividing at least one hash value into multiple sections of rim-keys (L, A) of a given bit-width; and - means for determining leave rim-keys (L) being archive leaves of the Sparsed Merkle tree, and parent rim keys (A) supporting the creation of a subtree (SMT).

9. System according to the preceding claim, comprising means for creating a subtree (SMT) on a parent rim-key (A) if an archived data has the same rim key as the parent rim-key.

10. System (1a) according to claim 8 or 9, comprising means for dividing multiple hash values, preferably every hash values, into multiple sections of rim-keys of the given bit-width.

11. System (1a) according to any of claims 8 to 10, wherein at least one subtree is a vector of hash of a given index.

12. System (1a) according to any of claims 8 to 11, wherein the index is the respective rim key.

13. System (1a) according to any of claims 8 to 12, wherein the Sparsed Merkle Tree comprises vectors of hashes.

14. Database (2a) for storing and keeping track of sets of data in a specific network, implementing a Sparsed Merkle tree, comprising - hash values of the data obtained by cryptographically hashing the data; - combined hash values of a first stage obtained by cryptographically hashing some of the consecutive hash values of the data; - combined hash values of a second stage obtained by cryptographically hashing some of the combined hash values of a first stage; ..., - a root hash value, **characterized by** - rim-keys of a given bit-width obtained by dividing at least one hash value into multiple sections of rim-keys; and among which are leave rim-keys being archive leaves of the Sparsed Merkle tree, and parent rim keys supporting the creation of a subtree.

15. Network comprising one or more central unit, in particular one or more computerized central unit (3a), and connection(s) (4a) to additional command units (5a) implementing transactions, in particular computerized command units, the central unit(s) (3a) comprising a system (1a) according to any of claims 8 to 13 and a database (2a) according to the preceding claim.

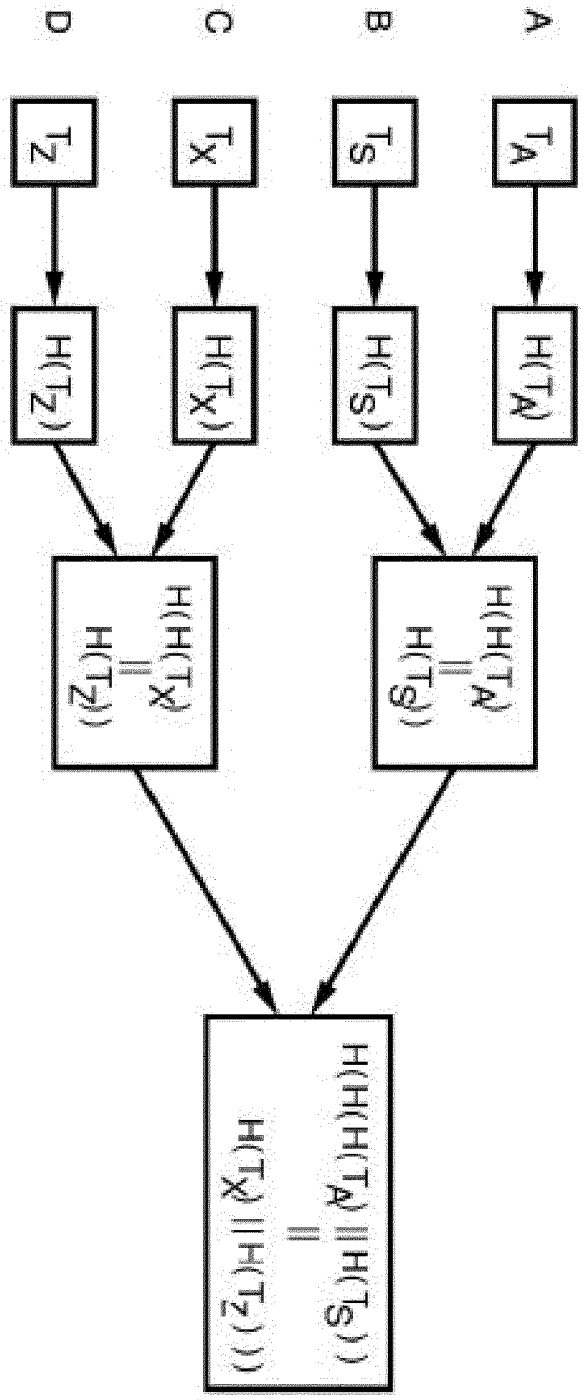
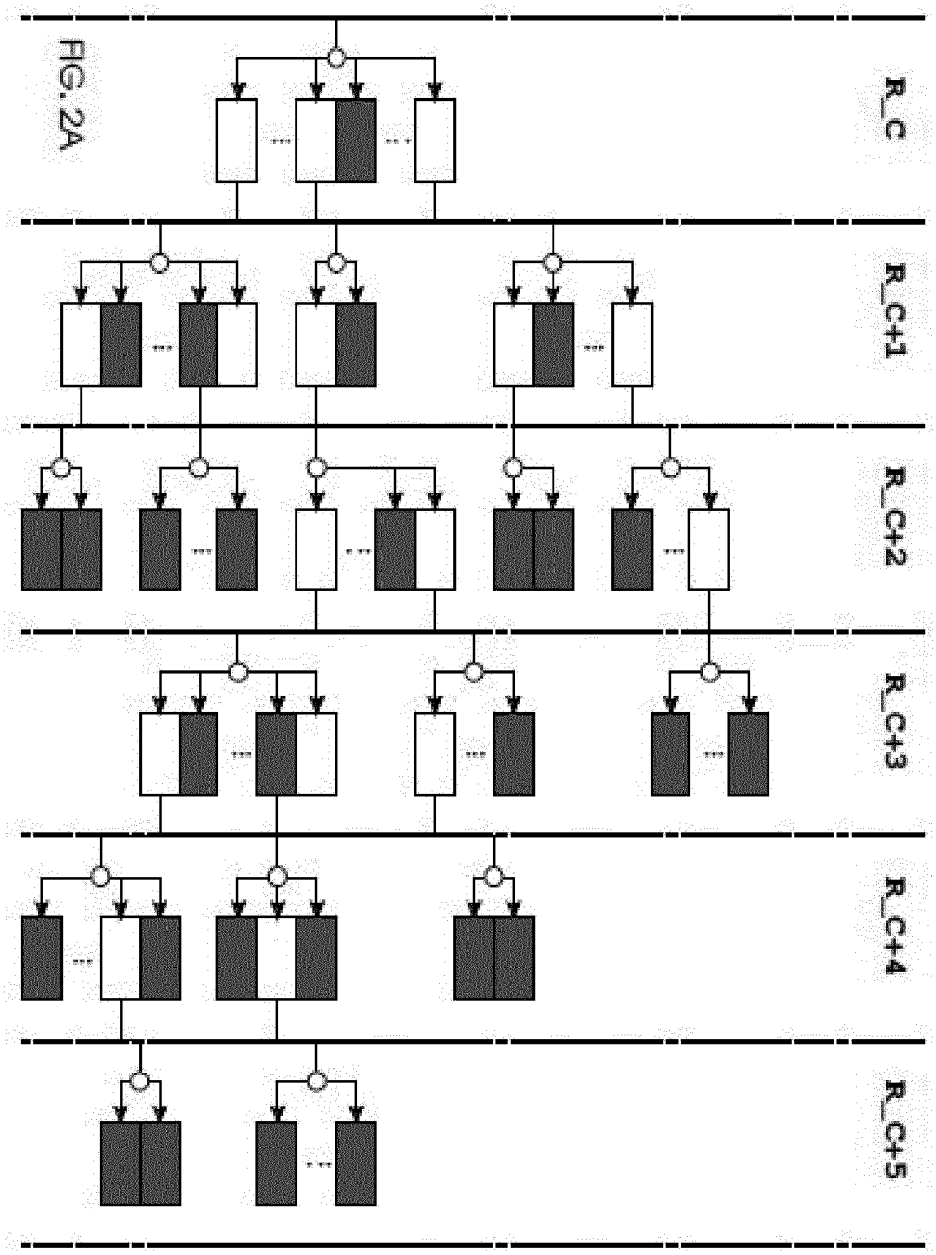


FIG. 1 (Prior art)

FIG. 2A



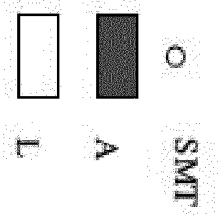
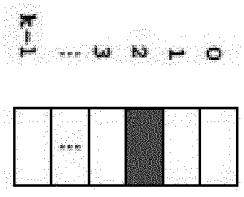


FIG. 2B



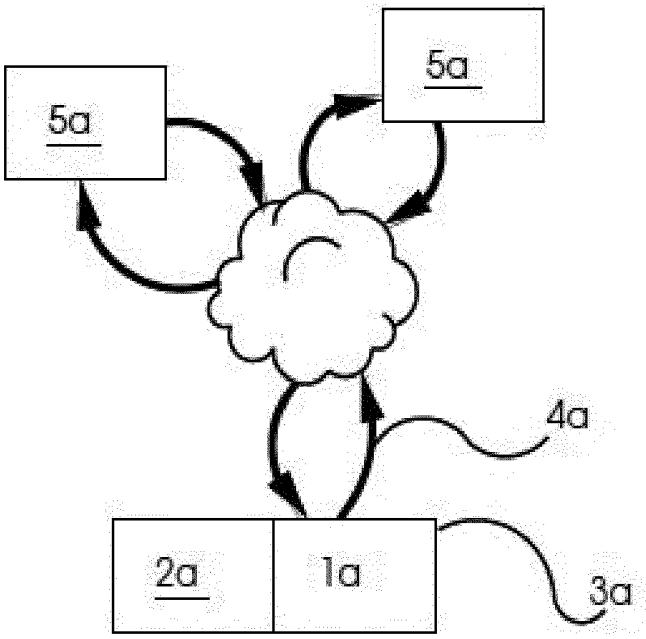


FIG. 2C

20	A3	**33**	49	··		
20	A3	57	**19**	94	··	
20	A3	57	**8A**	94	··	
20	A3	**C2**	C2	··		
20	A3	CA	48	**4A**	29	··
20	A3	CA	48	**7C**	73	··
20	A3	CA	48	**D9**	38	··
20	A3	C7	**9B**	38	··	

FIG. 3A

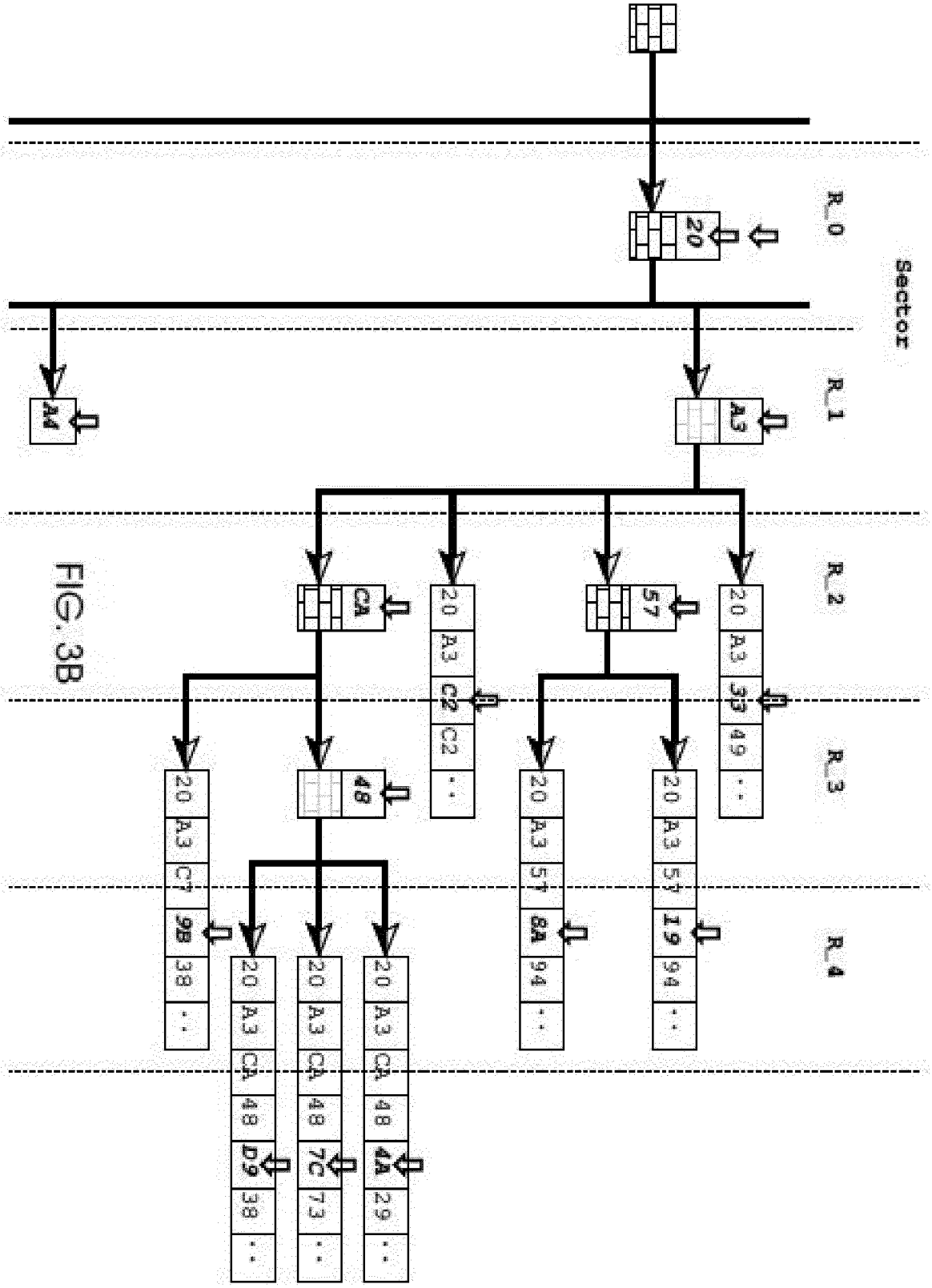
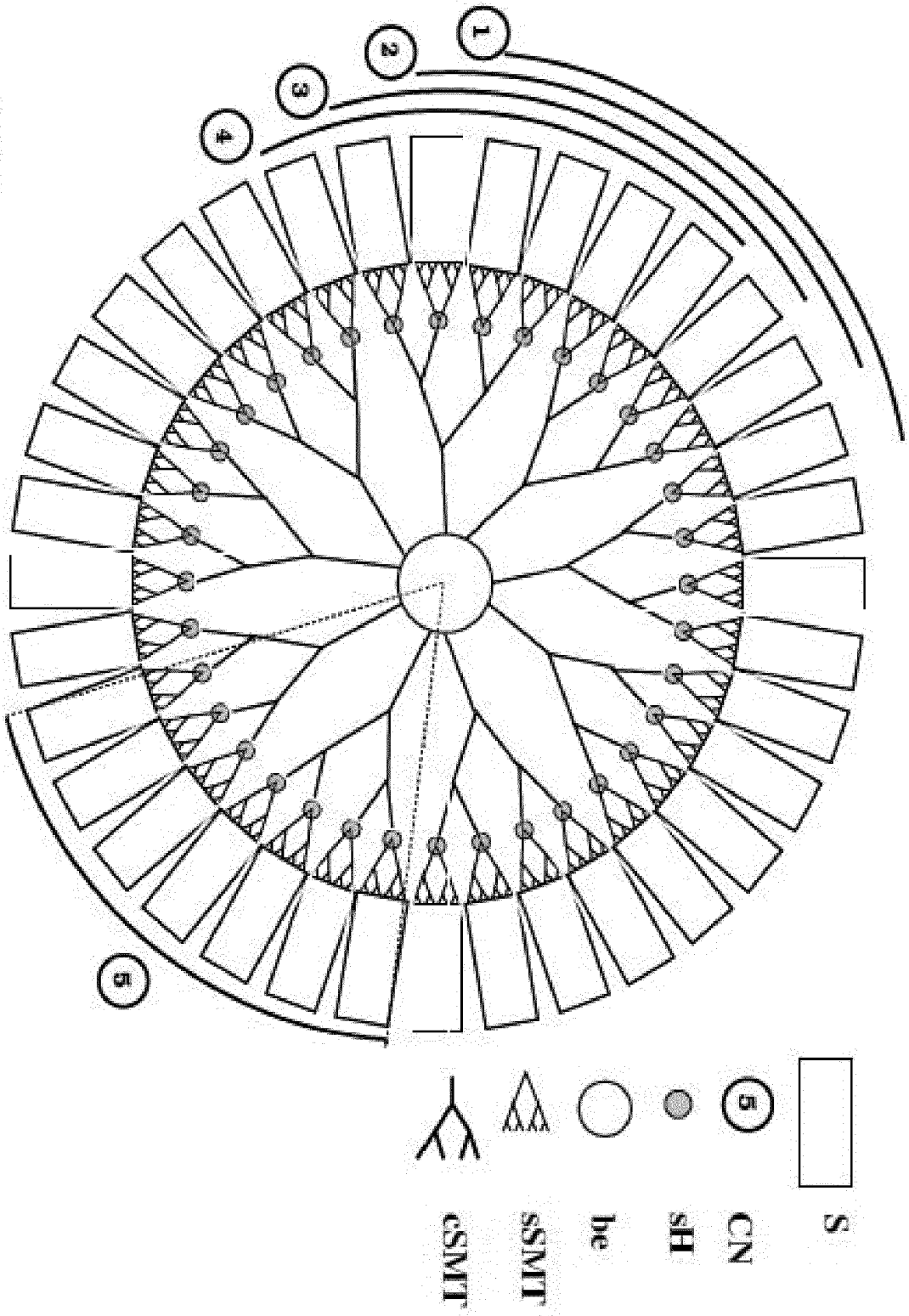

R_0
R_1
R_2
R_3
R_4
Sector
20
A4
A3
CA
20 A3 C2 C2 ..
57
20 A3 33 49 ..
20 A3 C7 9B 38 ..
48
20 A3 CA 48 D9 38 ..
20 A3 CA 48 7C 73 ..
20 A3 CA 48 4A 29 ..
20 A3 57 8A 94 ..
20 A3 57 19 94 ..

FIG. 3B

FIG. 4



Europäisches Patentamt
European Patent Office
Office européen des brevets

EUROPEAN SEARCH REPORT

Application Number
EP 19 02 0509

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
X	RASMUS DAHLBERG ET AL: "Efficient Sparse Merkle Trees: Caching Strategies and Secure (Non-)Membership Proofs", IACR, INTERNATIONAL ASSOCIATION FOR CRYPTOLOGIC RESEARCH, vol. 20160831:073914, 31 August 2016 (2016-08-31), pages 1-16, XP061021676, [retrieved on 2016-08-31] * sections 3 and 5.2; figures 2,3,6 *	1-15	INV. H04L9/32
A	US 2005/038774 A1 (LILLIBRIDGE MARK DAVID [US] ET AL) 17 February 2005 (2005-02-17) * the whole document *	1-15	

TECHNICAL FIELDS SEARCHED (IPC)
H04L

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
Munich	27 February 2020	Manet, Pascal

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 19 02 0509

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

27-02-2020

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2005038774 A1	17-02-2005	NONE	

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82