



(51) International Patent Classification:
G06F 17/30 (2006.01) *G06F 11/14* (2006.01)

(21) International Application Number:
PCT/EP2011/056067

(22) International Filing Date:
15 April 2011 (15.04.2011)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
1006442.6 17 April 2010 (17.04.2010) GB

(71) Applicant (for all designated States except US): **THE UNIVERSITY COURT OF THE UNIVERSITY OF EDINBURGH** [GB/GB]; Old College, Edinburgh EH8 9YL (GB).

(72) Inventors; and

(75) Inventors/Applicants (for US only): **ANDERSON, Paul** [GB/GB]; School of Informatics, Informatics House, 10 Crichton Street, Edinburgh EH8 9AB (GB). **WILKINSON, Simon** [GB/GB]; School of Informatics, Informatics Forum, 10 Crichton Street, Edinburgh EH8 9AB (GB).

(74) Agents: **HIGGIN, Paul** et al.; Swindell & Pearson Ltd, 48 Friar Gate, Derby, Derbyshire DE1 1GY (GB).

(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, 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, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, 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, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, 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, ML, MR, NE, SN, TD, TG).

Published:

— with international search report (Art. 21(3))

(54) Title: STORING A DIRECTORY TREE STRUCTURE

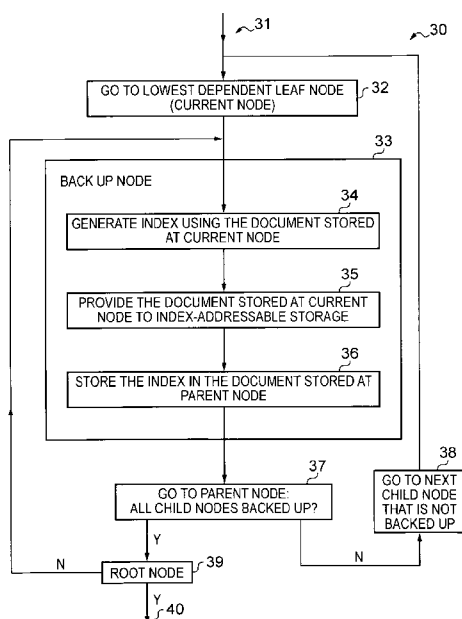


FIG. 4

(57) Abstract: A method comprising: descending a directory tree structure by identifying a leaf node of the directory tree structure that has not been backed up as a current node, wherein the current node has a current document that has not been backed up; and backing up the current node by: providing the current document to an index-addressable storage; and storing a current index for the current node in a parent document for a parent node of the current node wherein the current index for the current node has been generated using the current document and has a value that uniquely represents the content of the current document.

TITLE

Storing a directory tree structure

FIELD OF THE INVENTION

5

Embodiments of the present invention relate to methods, apparatus, computer programs for storing a directory tree structure and for retrieving a stored directory tree structure. In particular, they relate to back up and/or restore.

10 BACKGROUND TO THE INVENTION

A directory tree structure is a hierarchical organization of data structures typically comprising folders and files. A folder can depend from another folder but cannot depend from a file. A file can depend from a folder but cannot
15 depend from a file.

The directory tree structure forms an acyclic directed rooted tree graph with the files as leaf nodes and the folders as internal nodes.

20 It is desirable to copy a directory tree structure.

It is desirable to copy a directory tree structure to a remote storage facility to back up the directory tree structure and to enable restore of the directory tree structure at a later time.

25

It would be desirable to have a remote storage facility that can be used to store many directory tree structures for different users.

If two directory tree structures have identical sub-trees or identical files, it
30 would be desirable to avoid duplicating the back up of common sub-trees or files that can be backed up once and then shared.

2

BRIEF DESCRIPTION OF VARIOUS EMBODIMENTS OF THE INVENTION

According to various, but not necessarily all, embodiments of the invention there is provided a method comprising:

- 5 descending a directory tree structure by identifying a leaf node of the directory tree structure that has not been backed up as a current node, wherein the current node has a current document that has not been backed up; and
backing up the current node by:

- 10 providing the current document to an index-addressable storage; and
storing a current index for the current node in a parent document for a parent node of the current node wherein the current index for the current node has been generated using the current document and has a value that uniquely represents the content of the current document.

- 15 According to various, but not necessarily all, embodiments of the invention there is provided an apparatus comprising:

- means for descending a directory tree structure by identifying a leaf node of the directory tree structure that has not been backed up as a current node, wherein the current node has a current document that has not been backed
20 up; and
means for backing up the current node comprising:

- means for generating a current index for the current node using the current document, wherein the current index has a value that uniquely represents the content of the current document;
25 means for providing the current document to an index-addressable storage; and
means for storing the current index in a parent document for a parent node of the current node.

30

According to various, but not necessarily all, embodiments of the invention there is provided a storage system comprising:

3

one or more storage apparatus that stored a plurality of documents and a plurality of associated indexes, where each index is used to access a respective document and has a value that uniquely represents the content of the respective document; and

5 an interface configured to receive an index and to provide in reply a document associated with the index,

wherein a sub-set of the plurality of documents and the plurality of associated indexes define a directory tree having internal nodes and leaf nodes,

10 wherein any document of the sub-set comprising one or more indexes represents a respective internal node of the directory tree and the one or more indexes access documents representing child nodes of the respective internal node and

wherein documents of the sub-set comprising data and not comprising one or more indexes represent leaf nodes of the directory tree.

15

According to various, but not necessarily all, embodiments of the invention there is provided a method comprising:

decrypting an encrypted master document comprising an index representing a root node of a directory tree to obtain the index;

20 using the index to query a database storing a plurality of documents and a plurality of associated indexes, where each index is used to access a respective document and has a value that uniquely represents the content of the respective document;

25 wherein a sub-set of the plurality of documents and the plurality of associated indexes define the directory tree having internal nodes and leaf nodes, wherein any document of the sub-set comprising one or more indexes represents a respective internal node of the directory tree and the one or more indexes access documents representing child nodes of the respective internal node and

30 wherein documents of the sub-set comprising data and not comprising one or more indexes represent leaf nodes of the directory tree, and

4

receiving in reply a document of the sub-set comprising one or more indexes and representing a respective root node of the directory tree, wherein the one or more indexes access documents representing child nodes of the root node in the directory tree structure.

5

According to various, but not necessarily all, embodiments of the invention there is provided a method comprising:

identifying a node of a graph that represents a document that has not processed; and

10 processing the document by:

providing the document to an index-addressable storage; and

storing an index for the provided document in a further document represented by a parent node of the identified node in the graph wherein the index has been generated based on the provided

15 document and has a value that represents the content of the provided document.

According to various, but not necessarily all, embodiments of the invention there is provided an apparatus comprising:

20 means for identifying a node of a graph that represents a document that has not been processed; and

means for processing the document comprising:

means for providing the document to an index-addressable storage; and

25 means for storing an index for the provided document in a further document represented by a parent node of the identified node in the graph wherein the index has been generated based on the provided document and has a value that represents the content of the provided document.

30

According to various, but not necessarily all, embodiments of the invention there is provided a storage system comprising:

5

one or more storage apparatus that stored a plurality of documents and a plurality of associated indexes, where each index is used to access a respective document and has a value that uniquely represents the content of the respective document; and

5 an interface configured to receive an index and to provide in reply a document associated with the index,

wherein a sub-set of the plurality of documents and the plurality of associated indexes are represented by a graph having internal nodes and leaf nodes,

10 wherein any document of the sub-set comprising one or more indexes represents a respective internal node of the graph and the one or more indexes access documents representing child nodes of the respective internal node and

wherein documents of the sub-set comprising data and not comprising one or more indexes represent leaf nodes of the graph.

15

According to various, but not necessarily all, embodiments of the invention there is provided a method comprising:

decrypting an encrypted master document comprising an index representing a node of a graph to obtain the index;

20 using the index to query a database storing a plurality of documents and a plurality of associated indexes, where each index is used to access a respective document and has a value that uniquely represents the content of the respective document;

25 wherein a sub-set of the plurality of documents and the plurality of associated indexes are represented by the graph having internal nodes and leaf nodes, wherein any document of the sub-set comprising one or more indexes represents a respective internal node of the graph and the one or more indexes access documents representing child nodes of the respective internal node and

30 wherein documents of the sub-set comprising data and not comprising one or more indexes represent leaf nodes of the graph, and

6

receiving in reply a document of the sub-set comprising one or more indexes and representing a respective root node of the graph, wherein the one or more indexes access documents representing child nodes of the root node in the graph.

5

BRIEF DESCRIPTION OF THE DRAWINGS

For a better understanding of various examples of embodiments of the present invention reference will now be made by way of example only to the accompanying drawings in which:

10

Fig. 1 schematically illustrates a content-addressable storage (CAS) system;

Fig 2 schematically illustrates a directory tree structure;

Fig 3 schematically illustrates an alternative representation of the directory structure in a form in which it is stored in the content addressable storage

15

system;

Fig 4 schematically illustrates a method suitable for storing the directory structure in the content addressable storage system;

Fig 5 schematically illustrates an optional document encryption block;

Fig 6 illustrates one example of encryption;

20

Fig 7 schematically illustrates a security method;

Fig 8 schematically illustrates an example of a remote back up system; and

Fig 9 schematically illustrates a suitable controller for a host client apparatus or a host server apparatus.

25 DETAILED DESCRIPTION OF VARIOUS EMBODIMENTS OF THE INVENTION

Fig. 1 schematically illustrates a content-addressable storage (CAS) system

2. In this example, the content-addressable storage (CAS) system 2 is an

30

index addressable storage system in which a document dependent index 3 is used to reference a document 5.

7

The system 2 comprises: one or more storage apparatuses 4 and an interface 6.

5 The document 5 may be any collection of data whether structured or unstructured.

10 The storage apparatus or apparatuses 4 store a plurality of documents 5 and a plurality of associated indexes 3. Each index 3 is used to access a respective associated document 5 and has a value that represents the content of the respective associated document 5. The value uniquely represents the content at least with respect to the storage apparatus or apparatuses 4 and may, in some embodiments, be a globally unique representation of the content.

15 The interface 6 is configured to handle database queries in the form of a received index 3. If the received index 3 is associated in the storage apparatus 4 with a document 5 then that document 5 is returned via the interface 6. If the received index 3 is not associated in the storage apparatus 4 with a document 5 then a document 5 is not returned via the interface 6.

20

The interface 6 is configured to augment the database by storing a document 5 and its associated index 3 when received via the interface 6. Storage only occurs if the index 3 is not already associated with a document 5 in the storage apparatus 4. Alternatively, the system 2 may have an index generator (not shown) which would generate an index 3 from a received document 5. In this implementation, it is not necessary for the index 3 to be received by the interface 6.

25

30 The index 3 performs two functions. It operates as content-address in the CAS system 2 and it has a value that uniquely represents the content of the document from which it is formed and with which it is associated in the CAS system 2. This latter property of the index in combination with the fact that the

CAS system 2 only stores a document if its index is not already stored, prevents duplication of documents.

5 The index 3 may, for example be a cryptographic hash value produced by providing the document 5 as input to a cryptographic hash function. A cryptographic hash function is a deterministic procedure that takes an arbitrary block of data and returns a fixed-size bit string, the (cryptographic) hash value. Any change to the data will change the hash value.

10 Fig 2 schematically illustrates an example of a directory structure 10 for the storage and organization of data structures. The directory structure 10 comprises folders 11 and files 12. A folder can contain a file but a file cannot contain a folder. The directory structure forms a tree with a root folder 11₀₀.

15 The directory tree structure 10 may be represented as a directed acyclic (rooted tree) graph. The graph has internal nodes that have one or more dependent nodes. The root node is a special case of an internal node as it does not depend from another node. The graph has leaf nodes that depend from one or more internal nodes but that do not have dependent nodes.

20

A node X_{nm} in the graph is positioned at hierarchical level n ($n=0, 1, 2..$) and at lateral index m ($m=0, 1, 2, 3...$). The root node X_{00} is a folder ($X=11$) separately connected to two dependent child nodes- a first node X_{10} and a second node X_{11} via respective edges 13_{10} , 13_{11} . The first node X_{10} is an internal node (folder, $X=11$) and is separately connected to dependent child nodes X_{20} , X_{21} via respective edges 12_{20} , 12_{21} . The second node X_{10} is a leaf node (file, $X=12$) and is not connected to any dependent child nodes.

25

Fig 3 schematically illustrates an alternative representation of the directory tree structure 10 in a form in which it is stored in the content addressable storage system 2.

30

The directory tree structure 10 is represented as a directed acyclic (rooted tree) graph 20. The graph 20 has internal nodes that have one or more dependent nodes. The root node 21 is a special case of an internal node as it does not depend from another node. The graph 20 has leaf nodes that depend from one or more internal nodes but that do not have dependent nodes. A node that depends from another node is called a 'child' or 'dependent' node. A node from which one or more other nodes dependent is called a 'parent' node.

- 10 The graph 20 has a similar arrangement of nodes as the graph for the directory tree structure as illustrated in Fig 2. However, the graph 20 represents, at each node, a document 5.

15 A node 22_{nm} in the graph is positioned at hierarchical level n ($n=0, 1, 2..$) and at lateral index m ($m=1, 2, 3..$). The root node 22_{00} is separately connected to two dependent child nodes- a first node 22_{10} and a second node 22_{11} via respective edges. The first node 22_{10} is an internal node and is separately connected to dependent child nodes 22_{20} , 22_{21} via respective edges. The second node 22_{10} is a leaf node and is not connected to any dependent child nodes.

20 Each node 22_{nm} has a corresponding document 5_{nm} . If a node 22_{nm} is a leaf node the corresponding document 5_{nm} is the file 12_{nm} . If a node 22_{nm} is an internal node, the corresponding document 5_{nm} is a document that comprises one or more indexes 3. The one or more indexes 3 are usable to access documents at child nodes of the respective internal node. The one or more indexes 3 may, for example, be cryptographic hash values of the documents at child nodes of the respective internal node.

- 30 As an example, referring to Fig 3, the root node 22_{00} is an internal node, the corresponding document 5_{00} at this node comprises index 3_{10} and index 3_{11} .

10

Index 3_{10} is the cryptographic hash value of document 5_{10} which is stored at internal node 22_{10} . It can be used to access the document 5_{10} in the CAS system 2.

- 5 Index 3_{11} is the cryptographic hash value of document 5_{11} which is stored at leaf node 22_{11} . It can be used to access the document 5_{11} in the CAS system 2.

Fig 4 schematically illustrates a method suitable for storing the directory tree structure 10 in the content addressable system 2. The method 10 may be a back up method.

The method starts 21, then at block 32 the directory tree structure 10 is descended. A leaf node 22 of the directory tree structure 10 that has not been
15 backed up is identified as a 'current' node and its file as a current document 5.

A node 22 is backed up when the document 5 at the node is stored in the CAS system 2.

- 20 Next at block 33, the current node is backed up. This can be divided into a number of sub-blocks.

At sub-block 34, a current index 3 for the current node is generated using the current document 5. The current index 3 has a value that uniquely represents
25 the content of the current document 5. The current index 3 may, for example, be a cryptographic hash value of the current document 5.

At sub-block 35, the current document 5 is provided to an index-addressable storage 2. It is assumed in this implementation that the index-addressable
30 storage 2 is configured to generate an index 3 from the current document 5. If this were not the case, the index 3 could be sent with the current document 5.

11

At sub-block 36, the current index 3 is stored in a 'parent document', that is the document 5 at a parent node of the current node.

5 As an example, referring to Fig 3, the root node 22_{00} is an internal node, its corresponding document 5_{00} at this node comprises index 3_{10} for document 5_{10} at node 22_{10} and index 3_{11} for document 5_{11} at node 22_{11} .

10 Next the method moves to block 37, where the directory tree structure 10 is ascended one level by identifying the parent node of the current node as the new current node and the parent document as the new current document.

At block 37, if all the child nodes of the parent node have been backed up, then the parent document can be closed and the method moves to block 39.

15 At block 39, it is determined whether or not the current node is the root node. If it is the root node, the method ends 40. If it is not the root node, the current node is backed up at block 33 and the method continues as described above.

20 At block 37, if all the child nodes of the parent node have not been backed up, then the parent document remains open and the method moves to block 38.

25 At block 38, the directory tree structure 10 is traversed by identifying a child node of the current node that has not been backed up, as the new current node and returning to block 32. At block 32, the directory tree structure 10 is descended, if possible, to a leaf node 22 that directly or indirectly depends from the current node and the leaf node is identified as the new current node. The method then moves to block 33 to back up the leaf node.

30 It will therefore be appreciated, that ascending the directory tree structure only occurs when traversing is not or no longer available and block 37 exits to block 39 rather than block 38. After having ascended one level in the directory structure to the parent node and after backing up that parent node, traversing

12

the directory structure will occur, if available, (block 38) before further ascending up the directory structure (block 39).

5 The directory tree structure is, as described above, backed-up from the bottom up. Each document at an internal node inherits, via block 36, an index generated from each of its child node documents.

10 It should be appreciated that the method described requires the identification of at least leaf nodes 22 of the directory tree structure 10 that have not been backed up. This may be achieved by generating an index 3 for each node using its document. The index 3 has a value that uniquely represents the content of the document 5. The current index 3 may, for example, be a cryptographic hash value of the document. If the index 3 already exists as an index for the accessing a document in the CAS system 2, then it or an
15 equivalent has already been backed up at the CAS system 2. If the index 3 does not already exist as an index for accessing a document in the CAS system 2, then the node needs to be backed up.

20 Thus the step of generating an index 3 for a node illustrated in step 34 of Fig 4, may occur as prelude to the method 30 when the nodes that require back up are identified. If the indexes that are generated for identifying the nodes that require back up are cached, then they may be re-used later during back up. In this case, at step 34 of Fig 4, rather than generating the index, the step
25 would instead comprise obtaining the cached index.

The following pseudo code defines an alternative example of a back up method 30:

```
BackupNode(N) {  
30 If N is a directory, then let O = DirectoryObjectFor(N)  
Otherwise, let O = contents of N  
Let H = Hash(O)
```

13

if there is no item with index H in the backup store, then {

 Store O in the backup store with index H

 If N is a directory {

 For each entry E in the directory, BackupNode(E)

5 }

 }

}

DirectoryObjectFor(D) {

10 Create an empty directory object N

 For each entry E in the directory D {

 If E is a directory, then let O = DirectoryObjectFor(E)

 Otherwise, let O = contents of E

 Let H = Hash(O)

15 Add the metadata for E, and the hash H to N

 }

 Return N

}

20

Fig 5 schematically illustrates an optional document encryption block 34A, which may occur in the method 30 after block 34 and before block 35. At block 34A the current document is encrypted before providing the (encrypted) current document to the index-addressable storage at block 35.

25 Fig 6 illustrates one example of encryption. The document 5 is provided to a cryptographic hash algorithm which produces a hash value 43. The hash value 43 is used as a cryptographic key for encrypting the document. In the illustrated example, the document is provided to symmetric key encryption algorithm and the hash value is used as the symmetric key. The encrypted
30 document 5 is provided to a cryptographic hash algorithm which produces a hash value. This hash value is used as an index for the document.

The index and the cryptographic key are both stored in the parent document.

Additional properties of the document 5 may also be stored in the parent document. The properties may, for example, include metadata such as revision date.

Fig 7 schematically illustrates a method 50 that may be performed when the method 30 ends 40. The current node is a root node of the directory tree structure 10.

10

At block 51 a current index for the current node is generated using the current document. The current index may, for example, be a cryptographic hash value of the current document. The current index has a value that uniquely represents the content of the current document.

15

Next at block 52, the current document is provided to the index-addressable storage system 2.

Next at block 53, the current index is stored in a master document;

20

Next at block 54, the master document is encrypted using a secret. The secret is typically personal to the user like a unique ID or password.

Next at block 56, the encrypted master document is stored. The encrypted master document may be stored in an apparatus that hosts the directory structure 10, alternatively, the encrypted master document may be stored on a portable storage apparatus that can be carried by a user.

25

The method of restoring the directory tree structure 10 from back up comprises:

30

a) decrypting the encrypted master document comprising an index representing a root node of a directory tree to obtain the index;

15

b) using the index to query a database storing a plurality of documents and a plurality of associated indexes, where each index is used to access a respective document and has a value that uniquely represents the content of the respective document. A sub-set of the plurality of documents and the

5 plurality of associated indexes define a directory tree having internal nodes and leaf nodes. Any document of the sub-set comprising one or more indexes represents a respective internal node of the directory tree and the one or more indexes access documents representing child nodes of the respective internal node. Documents of the sub-set comprising data and not comprising
10 one or more indexes represent leaf nodes of the directory tree, and

c) receiving in reply a document of the sub-set comprising one or more indexes and representing a respective root node of the directory tree, wherein the one or more indexes access documents representing child nodes of the root node in the directory tree structure.

15

The indexes are used to retrieve further documents containing further indexes or files. The process of using recovered indexes to recover more documents which in turn provide more indices or files is repeated until the directory tree structure 10 is reproduced.

20

Fig 8 schematically illustrates an example of a remote back up system 60.

The system 60 comprises a server apparatus 64 connected to the content addressable storage system 2. The server apparatus 64 and the content
25 addressable storage system may be co-located at location 62 or located at different locations.

The system 60 also comprises a plurality of client apparatus 68A, 68B, 68C, 68D which connect to the server apparatus 64 via network 66. The network 66
30 may be a local area network, a metropolitan area network or a large area network for example. The network 66 may be comprised of a plurality of interconnected networks.

Each client 68 has a directory structure 10 and is configured to communicate with the server 64 to back up the directory server as described above.

- 5 The methods illustrated in Figs 4, 5, 6 and 7 (or equivalents or alternatives) may be performed at a client 48. The client 48, at each iteration of the back up block 33 in the method 30, sends the document stored at the current node to the server apparatus 64.
- 10 The client 48 is also configured to determine which nodes of the directory tree require backing-up.

It will be appreciated that if the client apparatuses 68A, 68B, 68C have sub-trees of their directory trees that are identical, then the document(s) used to
15 represent that sub-tree is only stored once rather than once per client.

The content addressable storage 2 comprises: one or more storage apparatus 4 (Fig 1) that store a plurality of documents 5 and a plurality of associated indexes 3, where each index is used to access a respective document and
20 has a value that uniquely represents the content of the respective document; and an interface 6 (Fig 1) configured to receive an index 3 and to provide in reply a document 5 associated with the index. A sub-set of the plurality of documents and the plurality of associated indexes define a directory tree having internal nodes and leaf nodes. Any document of the sub-set
25 comprising one or more indexes represents a respective internal node of the directory tree and the one or more indexes access documents representing child nodes of the respective internal node. Documents of the sub-set comprising data and not comprising one or more indexes represent leaf nodes of the directory tree.

30 An index for any document representing a node of the directory tree has a value that uniquely represents the structure and content of the sub-tree that

has the node as its root such that if the structure or content of the sub-tree where to change the document would change and the index would change. Furthermore, each document in the path to the root would change.

- 5 Fig 9 schematically illustrates a suitable controller for a host client apparatus 68 or a host server apparatus 64.

Implementation of controller can be in hardware alone (a circuit, a processor...), have certain aspects in software including firmware alone or
10 can be a combination of hardware and software (including firmware).

The controller may be implemented using instructions that enable hardware functionality, for example, by using executable computer program instructions in a general-purpose or special-purpose processor that may be stored on a
15 computer readable storage medium (disk, memory etc) to be executed by such a processor.

In the illustrated example, the controller comprises at least one processor 70 connected to read from and write to at least one memory 72. The memory 72
20 stores a computer program 74 comprising machine readable instructions which when loaded in to the processor control the operation of the host. The computer program 74 provides the logic and routines that enables the apparatus to perform the methods illustrated in any one or more of Figs 4, 5 , 6, 7. The processor 70 by reading the memory 72 is able to load and execute
25 the computer program 74.

The computer program may arrive at the memory 72 via any suitable delivery mechanism. The delivery mechanism may be, for example, a computer-readable storage medium, a computer program product, a memory device, a
30 record medium such as a CD-ROM or DVD, an article of manufacture that tangibly embodies the computer program 74. The delivery mechanism may be a signal configured to reliably transfer the computer program 74. The

controller may propagate or transmit the computer program 74 as a computer data signal.

Although the memory 72 is illustrated as a single component it may be
5 implemented as one or more separate components some or all of which may be integrated/removable and/or may provide permanent/semi-permanent/dynamic/cached storage.

The blocks illustrated in the Figs 4 to 7 may represent steps in a method
10 and/or sections of code in the computer program 74. The illustration of a particular order to the blocks does not necessarily imply that there is a required or preferred order for the blocks and the order and arrangement of the block may be varied. Furthermore, it may be possible for some steps to be omitted.

15 Reference has been made to cryptographic hash algorithms. Current suitable algorithms include MD5 and SHA-1.

Reference has been made to symmetric-key encryption algorithms. Current
20 suitable algorithms include AES-128, AES-256.

It would be a simple and straightforward modification to adapt the above described back up process (Fig 4) so that only modified sub-trees of the directory tree are processed. For example, the date/time of last back up may
25 be stored and compared against property values of the files and folders to determine which may be created or modified since the last back up. The nodes in the directory tree structure may be flagged and traversing the directory tree structure would be selective based on the flags. Block 37 would be modified to ask whether 'all flagged child nodes are backed up?'. Block 38
30 would be modified to 'go to the next flagged child node that is not backed up'. Block 32 would be modified to 'go to the lowest flagged dependent leaf node'.

19

Although embodiments of the present invention have been described in the preceding paragraphs with reference to various examples, it should be appreciated that modifications to the examples given can be made without departing from the scope of the invention as claimed.

5

Technically it is possible for two different documents to produce the same hash value. However, with a proper hash function the chances of such a hash collision are sufficiently small to be insignificant. A hash value may therefore be considered to be practically globally unique.

10

In some embodiments, a document may be broken-up into parts for back up and each part is then processed separately as a separate document. In this case, the document may be represented as a parent node and the parts of the documents represented as dependent nodes from the parent node..

15

It should be appreciated that the index does not have to be produced directly from a document.

Features described in the preceding description may be used in combinations other than the combinations explicitly described.

20

Although functions have been described with reference to certain features, those functions may be performable by other features whether described or not.

25

Although features have been described with reference to certain embodiments, those features may also be present in other embodiments whether described or not.

30

Whilst endeavoring in the foregoing specification to draw attention to those features of the invention believed to be of particular importance it should be understood that the Applicant claims protection in respect of any patentable

20

feature or combination of features hereinbefore referred to and/or shown in the drawings whether or not particular emphasis has been placed thereon.

I/we claim:

5

CLAIMS

1. A method comprising:

5 descending a directory tree structure by identifying a leaf node of the directory tree structure that has not been backed up as a current node, wherein the current node has a current document that has not been backed up; and backing up the current node by:

10 providing the current document to an index-addressable storage; and storing a current index for the current node in a parent document for a parent node of the current node wherein the current index for the current node has been generated using the current document and has a value that uniquely represents the content of the current document.

15 2. A method as claimed in claim 1, further comprising: traversing the directory tree structure by identifying as a current node a leaf node of the directory tree structure that directly or indirectly depends from the parent node and has not been backed up, wherein the current node has a current document that has not been backed up and backing up the current node.

20 3. A method as claimed in claim 1 or 2, further comprising: ascending the directory tree by identifying the parent node as the current node and the parent document as the current document and backing up the current node.

25 4. A method as claimed in claim 3, further comprising: ascending the directory tree structure only when traversing is not or no longer available.

5. A method as claimed in claim 3 or 4, further comprising: after ascending the directory tree a level, traversing the directory tree structure.

30 6. A method as claimed in any preceding claim, wherein the current index is a cryptographic hash value of the current document.

22

7. A method as claimed in any preceding claim, further comprising: encrypting the current document before providing the current document to the index-addressable storage.

5 8. A method as claimed in claim 7, comprising encrypting the current document using the current index.

9. A method as claimed in claim 8, comprising encrypting the current document using the current index as a symmetric key provided to a symmetric
10 key encryption algorithm that operates on the current document.

10. A method as claimed in any preceding claim comprising, when the current node is a root node of the directory tree structure:
15 generating a current index for the current node using the current document, wherein the current index has a value that uniquely represents the content of the current document;
providing the current document to the index-addressable storage;
storing the current index in a master document.

20

11. A method as claimed in claim 10, further comprising:
encrypting the master document using personal secret; and
storing encrypted master document.

25

12. A method as claimed in any preceding claim comprising: identifying a leaf node of the directory tree structure that has not been backed up by generating an index for the leaf node using a document of the leaf node, wherein the index has a value that uniquely represents the content of the document and
30 determining whether the

13. A method as claimed in any preceding claim comprising: identifying a sub-tree of the directory tree structure that has not been backed up by generating an index for a root node of the sub-tree using a document of the root node of the sub-tree, wherein the index has a value that uniquely represents the content of the document and determining that the generated index is new.

14. A method as claimed in any preceding claim comprising: identifying a sub-tree of the directory tree structure that has been backed up by generating an index for a root node of the sub-tree using a document of the root node of the sub-tree, wherein the index has a value that uniquely represents the content of the document and determining that the generated index is not newly generated.

15. A method comprising:
identifying a file that has not been backed up;
providing the file and index to an index-addressable storage;
storing the index in a document associated with the folder comprising the file in the directory

16. A computer program which when loaded into a processor instructs the processor to enable the method of any one of claims 1 to 15.

17. An apparatus comprising: at least one processor and at least one memory wherein the at least one memory stores computer program instructions which, when loaded into the at least one processor, enable the apparatus to perform the method of any one of claims 1 to 15.

18. An apparatus comprising:
means for descending a directory tree structure by identifying a leaf node of the directory tree structure that has not been backed up as a current node, wherein the current node has a current document that has not been backed up; and

24

means for backing up the current node comprising:

means for generating a current index for the current node using the current document, wherein the current index has a value that uniquely represents the content of the current document;

5 means for providing the current document to an index-addressable storage; and

means for storing the current index in a parent document for a parent node of the current node.

10 19. An apparatus comprising means for performing a method as claimed in any of claims 1 to 15.

20. An apparatus as claimed in claim 18 or 19 configured to operate as a client apparatus.

15

21. A storage system comprising:

one or more storage apparatus that stored a plurality of documents and a plurality of associated indexes, where each index is used to access a respective document and has a value that uniquely represents the content of

20 the respective document; and

an interface configured to receive an index and to provide in reply a document associated with the index,

wherein a sub-set of the plurality of documents and the plurality of associated indexes define a directory tree having internal nodes and leaf nodes,

25 wherein any document of the sub-set comprising one or more indexes represents a respective internal node of the directory tree and the one or more indexes access documents representing child nodes of the respective internal node and

30 wherein documents of the sub-set comprising data and not comprising one or more indexes represent leaf nodes of the directory tree.

25

22. A storage system as claimed in claim 21, wherein an index for any document representing a node of the directory tree has a value that uniquely represents the structure and content of the sub-tree that has the node as its root such that if the structure or content of the sub-tree were to change the document would change and the index would change.

23. A method comprising:

decrypting an encrypted master document comprising an index representing a root node of a directory tree to obtain the index;

10 using the index to query a database storing a plurality of documents and a plurality of associated indexes, where each index is used to access a respective document and has a value that uniquely represents the content of the respective document;

wherein a sub-set of the plurality of documents and the plurality of associated
15 indexes define the directory tree having internal nodes and leaf nodes,

wherein any document of the sub-set comprising one or more indexes represents a respective internal node of the directory tree and the one or more indexes access documents representing child nodes of the respective internal node and

20 wherein documents of the sub-set comprising data and not comprising one or more indexes represent leaf nodes of the directory tree, and

receiving in reply a document of the sub-set comprising one or more indexes and representing a respective root node of the directory tree, wherein the one or more indexes access documents representing child nodes of the root node

25 in the directory tree structure.

24. A method comprising:

identifying a node of a graph that represents a document that has not processed; and

30 processing the document by:

providing the document to an index-addressable storage; and

26

storing an index for the provided document in a further document represented by a parent node of the identified node in the graph wherein the index has been generated based on the provided document and has a value that represents the content of the provided document.

25. A method as claimed in claim 24, wherein the graph is an acyclic graph.

10 26. A method as claimed in any one of claims 24 to 25 comprising identifying a node of the graph representing a document that has not been processed by generating an index for the node using a document represented by the node, wherein the index has a value that represents the content of the document and determining that the generated index is new.

15 27. A method as claimed in any one of claims 24 to 26 comprising: identifying a sub-graph of the graph representing a document that has not been processed by generating an index for an initial node of the sub-graph using an initial document represented by the initial node of the sub-graph, wherein the
20 index has a value that represents the content of the initial document and determining that the generated index is new.

28. A method as claimed in any one of claims 24 to 27 comprising identifying
25 all the sub-graphs of at least the sub-graph representing documents that have not been processed by generating an index for each initial node of the sub-graphs using an initial document represented by the respective initial nodes of the sub-graphs, wherein the index has a value that represents the content of the initial documents and determining that the generated index is new.

30 29. A method as claimed in any one of claims 24 to 28, comprising performing an ordered processing of the documents, represented by nodes of the graph, that have not been processed by, for each document,

27

providing the document to an index-addressable storage; and
storing an index for the provided document in a further document
represented by a parent node of the node representing the provided
document wherein the index has been generated based on the
5 provided document and has a value that represents the content of the
provided document.

30. A method as claimed in claim 29, wherein ordered processing of the
documents that have not been processed requires that a document
10 represented by a parent node in the graph is processed only after all of the
documents that are represented by dependent nodes of the parent node have
been processed,

31. A method as claimed in any one of claims 24 to 30 comprising identifying
15 a sub-graph of the graph that has been processed by generating an index for
an initial node of the sub-graph using an initial document represented by the
initial node of the sub-graph, wherein the index has a value that represents
the content of the initial document and determining that the generated index is
not new.

20

32. A method as claimed in any one of claims 24 to 32, wherein if it is not
possible to process the document by storing the index for the provided
document in a further document represented by a parent node of the identified
node in the graph, then storing the index for the provided document in a
25 master document.

33. A method as claimed in claim 32, further comprising:
encrypting the master document using a personal secret; and
storing the encrypted master document.

30

34. An apparatus comprising:

28

means for identifying a node of a graph that represents a document that has not been processed; and

means for processing the document comprising:

means for providing the document to an index-addressable storage;

5 and

means for storing an index for the provided document in a further document represented by a parent node of the identified node in the graph wherein the index has been generated based on the provided document and has a value that represents the content of the provided document.

10

35. An apparatus comprising means for performing a method as claimed in any of claims 24 to 33.

15 36. An apparatus as claimed in claim 34 or 35 configured to operate as a client apparatus.

37. A storage system comprising:

one or more storage apparatus that stored a plurality of documents and a

20 plurality of associated indexes, where each index is used to access a respective document and has a value that uniquely represents the content of the respective document; and

an interface configured to receive an index and to provide in reply a document associated with the index,

25 wherein a sub-set of the plurality of documents and the plurality of associated indexes are represented by a graph having internal nodes and leaf nodes, wherein any document of the sub-set comprising one or more indexes represents a respective internal node of the graph and the one or more indexes access documents representing child nodes of the respective internal node and

30

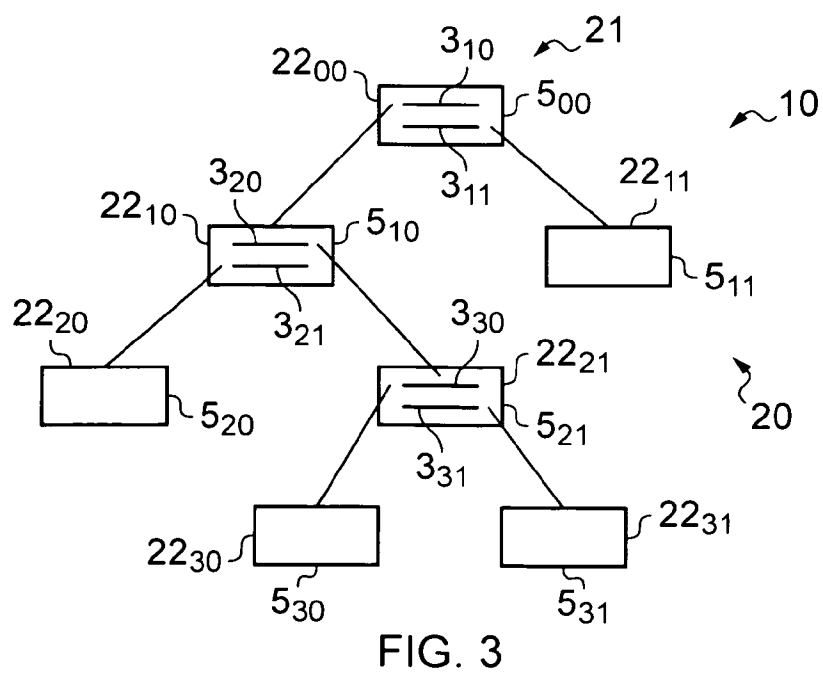
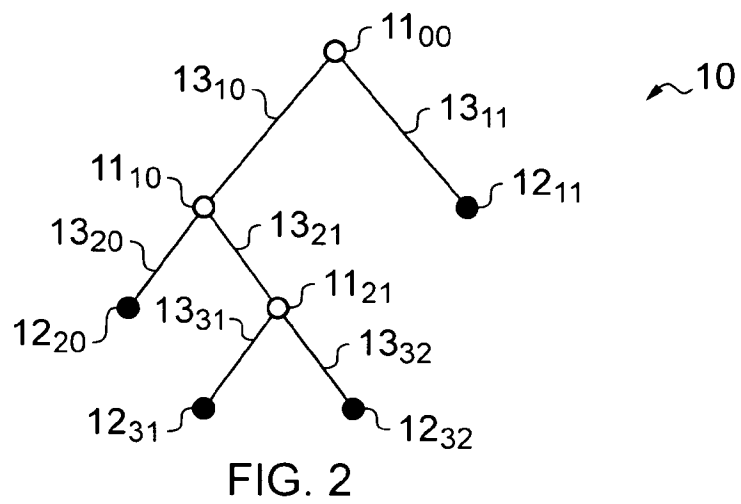
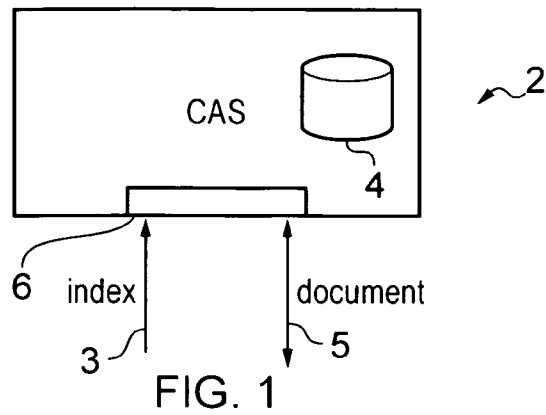
wherein documents of the sub-set comprising data and not comprising one or more indexes represent leaf nodes of the graph.

38. A storage system as claimed in claim 37, wherein an index for any document representing a node of the graph has a value that uniquely represents the structure and content of the sub-graph that has the node as its root such that if the structure or content of the sub-graph where to change the document would change and the index would change.

39. A method comprising:

decrypting an encrypted master document comprising an index representing a node of a graph to obtain the index;
using the index to query a database storing a plurality of documents and a plurality of associated indexes, where each index is used to access a respective document and has a value that uniquely represents the content of the respective document;
wherein a sub-set of the plurality of documents and the plurality of associated indexes are represented by the graph having internal nodes and leaf nodes, wherein any document of the sub-set comprising one or more indexes represents a respective internal node of the graph and the one or more indexes access documents representing child nodes of the respective internal node and
wherein documents of the sub-set comprising data and not comprising one or more indexes represent leaf nodes of the graph, and
receiving in reply a document of the sub-set comprising one or more indexes and representing a respective root node of the graph, wherein the one or more indexes access documents representing child nodes of the root node in the graph.

1/4



2/4

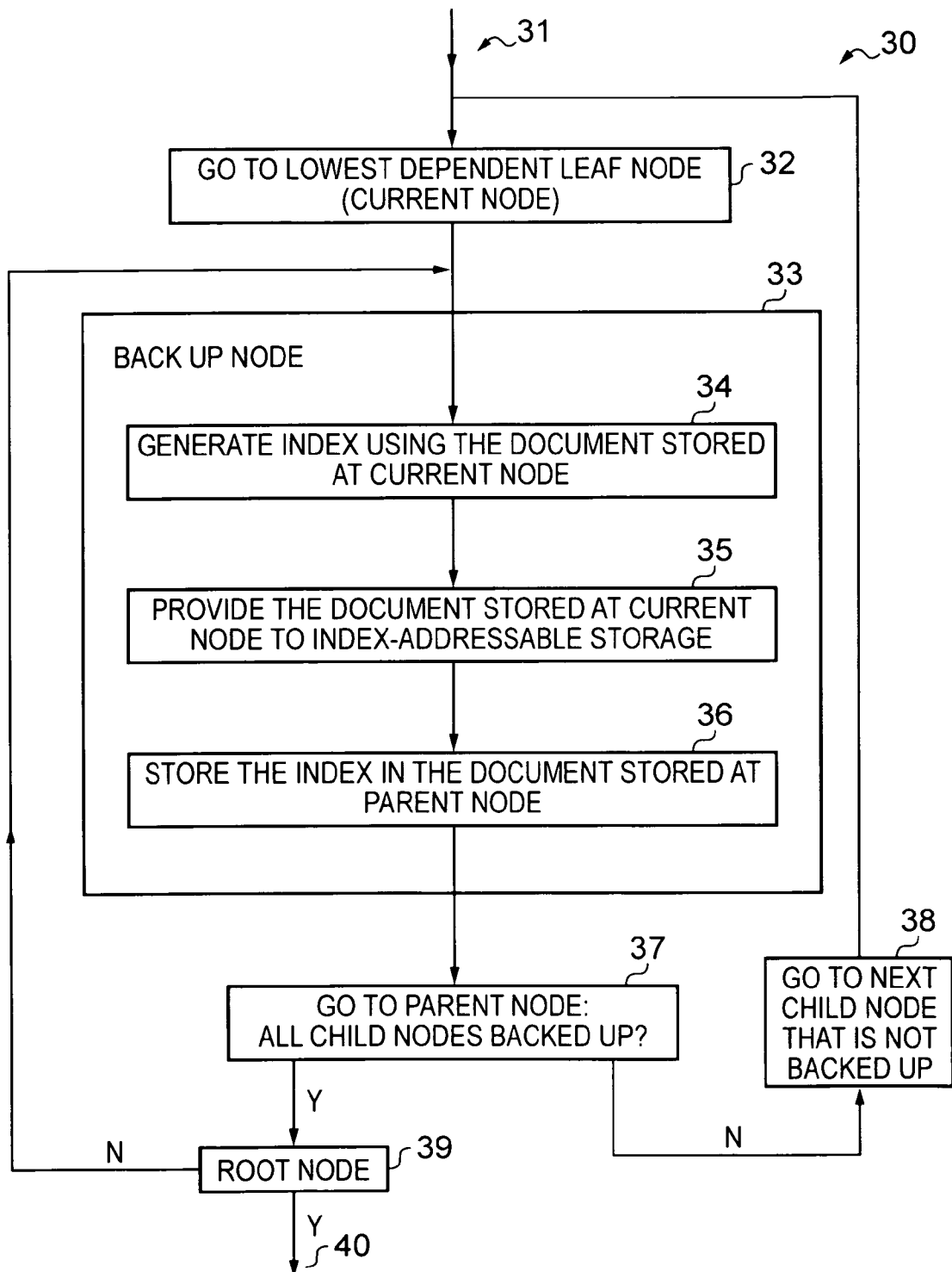


FIG. 4

3/4

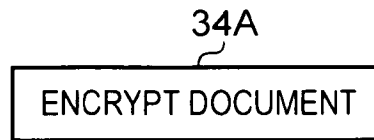


FIG. 5

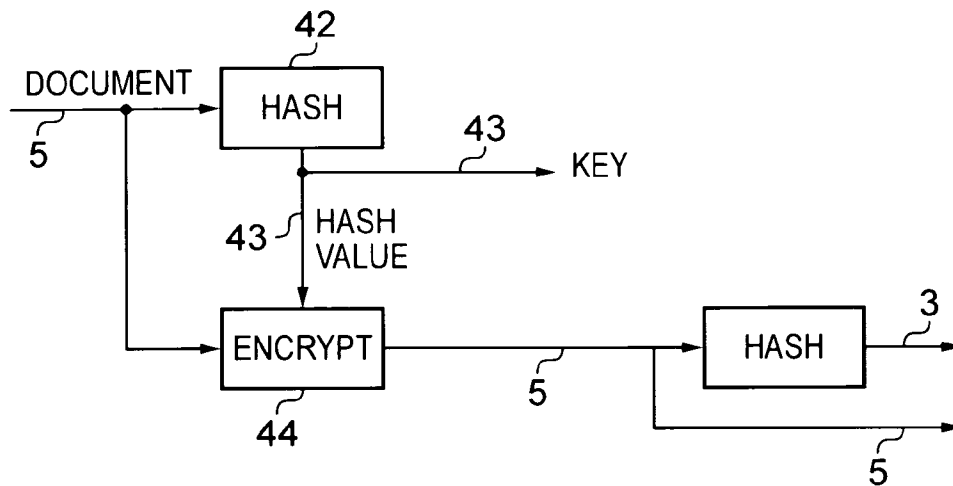


FIG. 6

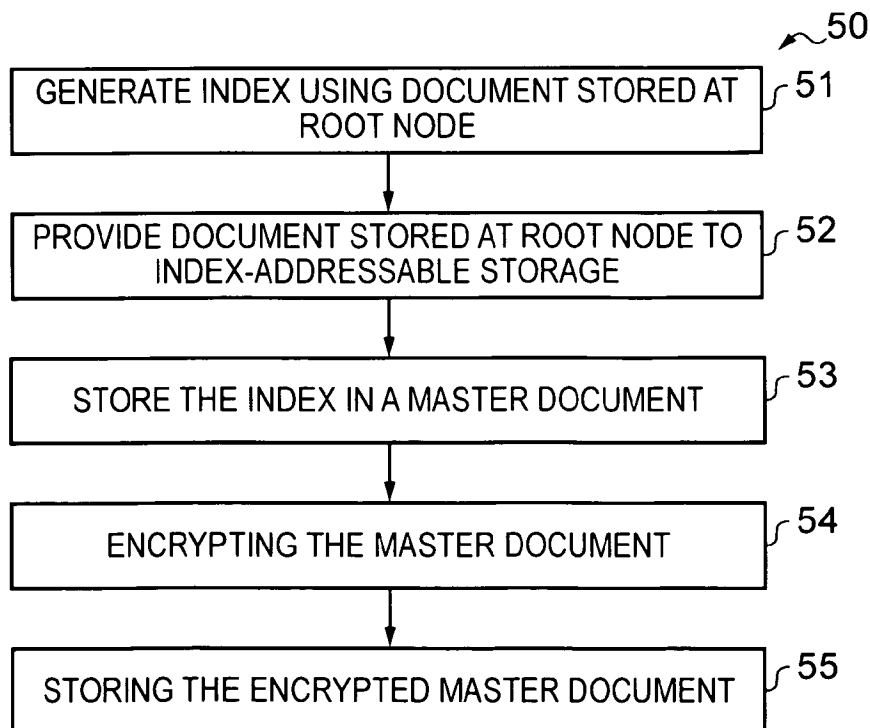


FIG. 7

4/4

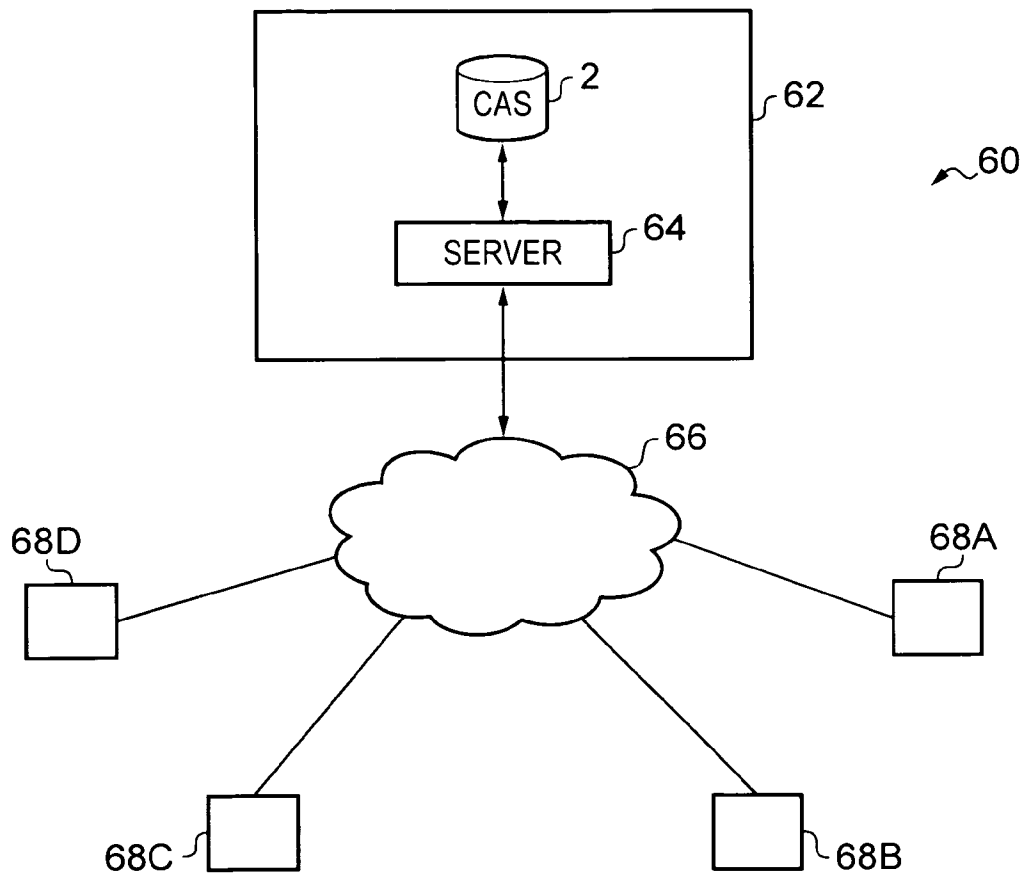


FIG. 8

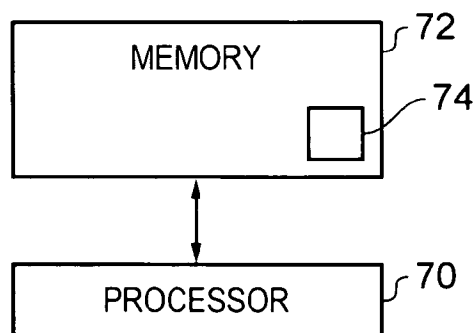


FIG. 9

INTERNATIONAL SEARCH REPORT

International application No

PCT/EP2011/056067

A. CLASSIFICATION OF SUBJECT MATTER

INV. G06F17/30 G06F11/14
ADD.

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F

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

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

EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2008/104146 A1 (SCHWAAB DAVID [US] ET AL) 1 May 2008 (2008-05-01) the whole document	1-39
A	US 2008/313236 A1 (VIJAYAKUMAR BALA [US] ET AL) 18 December 2008 (2008-12-18) the whole document	1-39
A	US 2007/112895 A1 (AHRENS MATTHEW A [US] ET AL) 17 May 2007 (2007-05-17) abstract; figure 4 paragraphs [0030] - [0032], [0036] - [0044]	1-39
A	WO 2006/039492 A2 (EMC CORP [US]) 13 April 2006 (2006-04-13) the whole document	1-39



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search

12 July 2011

Date of mailing of the international search report

22/07/2011

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040,
Fax: (+31-70) 340-3016

Authorized officer

Leuridan, Koen

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/EP2011/056067

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2008104146 A1	01-05-2008	DE 112008002947 T5 US 2010174684 A1 WO 2009059183 A1	02-09-2010 08-07-2010 07-05-2009
US 2008313236 A1	18-12-2008	NONE	
US 2007112895 A1	17-05-2007	NONE	
WO 2006039492 A2	13-04-2006	CN 101031908 A EP 1805649 A2 JP 2008515111 A US 2006085481 A1	05-09-2007 11-07-2007 08-05-2008 20-04-2006