US Patent & Trademark Office
Patent Public Search | Text View

United States Patent Application Publication	20250252163
Kind Code	A1
Publication Date	August 07, 2025
Inventor(s)	WRIGHT; CRAIG STEVEN et al.

METHOD AND SYSTEM FOR SECURING COMPUTER SOFTWARE USING A DISTRIBUTED HASH TABLE AND A BLOCKCHAIN

Abstract

A computer-implemented method and system for determining a metadata M for securing a controlled digital resource such as computer software using a distributed hash table and a peer-to-peer distributed ledger. This is a blockchain such as the Bitcoin blockchain. The method includes determining a data associated with the computer software and determining a first hash value based on the computer software. A second hash value based on the data and the computer software may be determined. The method further includes sending **140**, over a communications network, the data, the first hash value and the second hash value to an entry for storage in a distributed hash table. The second hash value may be a key of a key-value pair. The data and the first hash value may be a value in the key-value pair. A metadata (M) that is based on the second hash value may be determined for storage on the peer-to-peer distributed ledger.

Inventors: **WRIGHT; CRAIG STEVEN (LONDON, GB), SAVANAH; STEPHANE (LONDON, GB)**

Applicant: **nChain Licensing AG** (ZUG, CH)

Family ID: **58108696**

Appl. No.: **19/047260**

Filed: **February 06, 2025**

Foreign Application Priority Data

GB	1603125.4	Feb. 23, 2016
GB	1607058.3	Apr. 22, 2016

Related U.S. Application Data

parent US continuation 17401532 20210813 parent-grant-document US 12248539 child US 19047260
parent US continuation 16079074 20180822 parent-grant-document US 11455378 WO continuation PCT/IB2017/050827 20170214 child US 17401532

Publication Classification

Int. Cl.: **G06F21/12** (20130101); **G06F21/10** (20130101); **G06Q20/12** (20120101); **G06Q20/38** (20120101); **H04L9/00** (20220101); **H04L9/06** (20060101); **H04L9/32** (20060101)

U.S. Cl.:

CPC **G06F21/12** (20130101); **G06F21/10** (20130101); **G06Q20/123** (20130101); **G06Q20/1235** (20130101); **G06Q20/3823** (20130101); **G06Q20/3825** (20130101); **G06Q20/3827** (20130101); **G06Q20/3829** (20130101); **H04L9/0637** (20130101); **H04L9/0643** (20130101); **H04L9/3213** (20130101); **H04L9/3236** (20130101); **H04L9/3247** (20130101); **H04L9/3252** (20130101); G06Q2220/00 (20130101); H04L9/50 (20220501); H04L2209/56 (20130101)

Background/Summary

RELATED APPLICATIONS [0001] This application is a continuation application claiming priority from U.S. patent application Ser. No. 17/401,532, filed 13 Aug. 2021; which is a continuation application of U.S. patent application Ser. No. 16/079,074, filed 22 Aug. 2018, now U.S. Pat. No. 11,455,378, issued on 27 Sep. 2022; which is a U.S. National Stage under 35 USC 371 patent application, claiming priority to Serial No. PCT/IB2017/050827, filed on 14 Feb. 2017; which claims priority from GB 1607058.3, filed 22 Apr. 2016, and GB 1603125.4, filed 23 Feb. 2016, the entirety of all which is incorporated herein by reference.

TECHNICAL FIELD
[0002] The present disclosure relates to blockchain technology, security mechanisms and asset transfer. It is particularly suited for use as a mechanism for securing digital assets such as computer software and authorising/controlling access to the asset (eg computer software) using a distributed hash table and a peer-to-peer distributed ledger (eg blockchain).
BACKGROUND
[0003] In this document we use the term ‘blockchain’ to include all forms of electronic, computer-based, distributed ledgers. These include, but are not limited to, consensus-based blockchain and transaction-chain technologies, permissioned and un-permissioned ledgers, shared ledgers and variations thereof. The most widely known application of blockchain technology is the Bitcoin ledger, although other blockchain implementations have been proposed and developed. While Bitcoin may be referred to herein for the purpose of convenience and illustration, it should be noted that the invention is not limited to use with the Bitcoin blockchain and alternative blockchain implementations and protocols fall within the scope of the present invention.
[0004] A blockchain is a peer-to-peer, electronic ledger which is implemented as a computer-based decentralised, distributed system made up of blocks which in turn are made up of transactions. Each transaction is a data structure that encodes the transfer of control of a digital asset between participants in the blockchain system, and includes at least one input and at least one output. Each block contains a hash of the previous block to that blocks become chained together to create a permanent, unalterable record of all transactions which have been written to the blockchain since

its inception. Transactions contain small programs known as scripts embedded into their inputs and outputs, which specify how and by whom the outputs of the transactions can be accessed. On the Bitcoin platform, these scripts are written using a stack-based scripting language.
[0005] In order for a transaction to be written to the blockchain, it must be "validated". Network nodes (miners) perform work to ensure that each transaction is valid, with invalid transactions rejected from the network. Software clients installed on the nodes perform this validation work on an unspent transaction (UTXO) by executing its locking and unlocking scripts. If execution of the locking then unlocking scripts evaluates to TRUE, the transaction is valid and the transaction is written to the blockchain. Thus, in order for a transaction to be written to the blockchain, it must be i) validated by the first node that receives the transaction—if the transaction is validated, the node relays it to the other nodes in the network; and ii) added to a new block built by a miner; and iii) mined, i.e. added to the public ledger of past transactions.
[0006] Although blockchain technology is most widely known for the use of cryptocurrency implementation, digital entrepreneurs have begun exploring the use of both the cryptographic security system Bitcoin is based on and the data that can be stored on the Blockchain to implement new systems. It would be highly advantageous if the blockchain could be used for automated tasks and processes which are not limited to the realm of cryptocurrency. Such solutions would be able to harness the benefits of the blockchain (e.g. a permanent, tamper-proof record of events, distributed processing, etc) while being more versatile in their applications.
[0007] One area of current research is the use of the blockchain for the implementation of "smart contracts". These are computer programs designed to automate the execution of the terms of a machine-readable contract or agreement. Another area of blockchain-related interest is the use of 'tokens' (or 'coloured coins') to represent and transfer real-world entities via the blockchain. A potentially sensitive or secret item can be represented by the token which has no discernable meaning or value. The token thus serves as an identifier that allows the real-world item to be referenced from the blockchain, providing enhanced security.
[0008] It would be advantageous to be able to use security-related features such as cryptography and blockchain technologies for the purpose of transmitting, sharing or controlling access to digital assets such as, for example, computer software. Traditional approaches to securing the integrity and sharing of computer software involve the digital signing of the executables of the computer software. For instance, signing the executable or other associated code with a cryptographic pair of keys, such as a public key and a private key. The public key is often obtained from a trusted central authority such as a certification authority.
[0009] Computer software is often accompanied by a licence containing contractual obligations. The licence may contain the terms that govern the use or redistribution of the software. An issue may arise where the computer software or the associated licence is unlawfully transferred to another user.
[0010] Any discussion of documents, acts, materials, devices, articles or the like which have been included in the present specification is not to be taken as an admission that any or all of these matters form part of the prior art base or were common general knowledge in the field relevant to the present disclosure as it existed before the priority date of each claim of this application.
[0011] Throughout this specification the word "comprise", or variations such as "comprises" or "comprising", will be understood to imply the inclusion of a stated element, integer or step, or group of elements, integers or steps, but not the exclusion of any other element, integer or step, or group of elements, integers or steps.

SUMMARY

[0012] Embodiments of the invention may comprise a method and corresponding system for controlling access to, and/or transmission of, a controlled digital resource or asset. The invention may comprise a computer-implemented method for determining a metadata (M) for securing a controlled digital resource using a distributed hash table and a peer-to-peer distributed ledger (e.g.

blockchain). It may be described as a security method/system or a control method/system. It may be described as a method/system for securing the integrity of, control of and/or access to the controlled digital resource. The invention may comprise an authentication or authorisation method/system.
[0013] In one embodiment, the controlled digital resource may be computer software. Hereafter, the term “software” or “computer software” may be used instead of “controlled digital asset”.
[0014] The method may comprise the steps of: [0015] determining a data (D1) associated with the computer software; determining a first hash value (H1) of the computer software; determining a second hash value (H2) based on the data (D1) and the computer software; sending, over a communications network, the data (D1), the first hash value (H1) and the second hash value (H2) to an entry for storage in a distributed hash table, wherein the second hash value (H2) is a key of a key-value pair and the data (D1) and the first hash value (H1) are a value in the key-value pair; and determining a metadata (M) comprising the second hash value (H2) for storage on the peer-to-peer distributed ledger.
[0016] The method may further comprise determining a first redeem script (RS1), wherein the first redeem script is based on the metadata (M); and an agent public key (PA) associated with an agent (A). The redeem script may be a redeem script for a blockchain transaction (Tx).
[0017] The method may further comprise sending, over the communications network, a first data output (O1) for storage on the peer-to-peer distributed ledger based on an indication of a first quantity of cryptocurrency (C1) to be transferred, wherein the first quantity of cryptocurrency (C1) is associated with the first redeem script (RS1); and the first redeem script (RS1).
[0018] In the method, the data (D1) may comprise a licence associated with the computer software. The licence may be associated with a first user (U1) or a second user (U2) and further comprise a first user public key (PU1) associated with the first user (U1) or a second user public key (PU2) associated with the second user (U2). The licence may further comprise a hash value associated with at least one electronic device of the first user (U1) or the second user (U2). The licence may further comprise the first hash value (H1).
[0019] In the method, the second hash value (H2) may comprise a top hash value of a Merkle tree.
[0020] The invention may provide a computer-implemented method for authorising access to the computer software for a first user (U1), the method comprising determining a metadata (M) for securing a computer software according to the method described above; determining a second redeem script (RS2), wherein the second redeem script (RS2) is based on: the metadata (M); the agent public key (PA) associated with the agent (A); and the first user public key (PU1) associated with the first user (U1); sending, over a communications network, a second data output (O2) to the peer-to-peer distributed ledger based on: an indication that the first quantity of cryptocurrency (C1) from the first data output (O1) is to be transferred; and the second redeem script (RS2).
[0021] The method may further comprise determining an identifier indicative of the location of the computer software or the licence; assigning the identifier to the value in the key-value pair; and sending, over the communications network, the identifier to the entry on the distributed hash table.
[0022] A method for determining the location of the computer software or licence, the method comprising: determining a metadata (M) for securing a computer software according to the method described above; determining an identifier indicative of the location of the computer software or the licence; assigning the identifier to the value in the key-value pair; sending, over the communications network, the identifier to the entry on the distributed hash table; determining the metadata (M) from the first redeem script (RS1); retrieving the second hash value (H2) from the metadata (M); sending, over the communications network, the second hash value (H2) to a processor associated with a participating node of the distributed hash table; and determining, from the processor of the participating node, an identifier indicative of the location of the computer software or licence.
[0023] In the methods described above, the cryptocurrency may be Bitcoin and the peer-to-peer

distributed ledger may be the Bitcoin Blockchain.
[0024] Embodiments of the invention may comprise the step of providing metadata in a redeem script at a location which is designated in a blockchain protocol as a location for a cryptographic key.
[0025] One or more embodiments of the invention may comprise a method of embedding metadata in a blockchain transaction, substantially as described in the section below entitled “metadata”. This may comprise the steps of:
[0026] generating a blockchain transaction (Tx) having an output (TxO) related to a digital asset and a hash of a redeem script which comprises: [0027] metadata comprising a token which is a representation of, or a reference to, a tokenised entity; and [0028] at least one public cryptographic key.
[0029] The digital asset may be a quantity of cryptocurrency. The metadata may be provided in the redeem script at a location which is designated in a blockchain protocol as a location for a cryptographic key. The transaction Tx may be submitted to a blockchain network.
[0030] A computer software program comprising machine-readable instructions to cause a processing device to implement the methods described above.
[0031] A computer system for determining a metadata (M) for securing computer software using a distributed hash table, the system comprising a processing device associated with a node, configured to: determine a data (D1) associated with the computer software; determine a first hash value (H1) of the computer software; determine a second hash value (H2) based on the data (D1) and the computer software; send, over a communications network, the data (D1), the first hash value (H1) and the second hash value (H2) to an entry for storage in a distributed hash table, wherein the second hash value (H2) is assigned to a key of a key-value pair and the data (D1) and the first hash value (H1) are a value in the key-value pair; and determine a metadata (M) comprising the second hash value (H2).

Description

BRIEF DESCRIPTION OF DRAWINGS
[0032] FIG. **1** illustrates an example of a hash table.
[0033] Examples of the present disclosure will be described with reference to;
[0034] FIG. **2** illustrates a schematic diagram of an example system for determining a metadata (M) for securing computer software using a distributed hash table;
[0035] FIG. **3** illustrates a flow chart of a computer-implemented method for determining a metadata (M) for securing a computer software using a distributed hash table;
[0036] FIG. **4** illustrates an example of a Merkle tree;
[0037] FIG. **5** illustrates an example of a Merkle tree with reference to a computer software and a licence associated with a computer software;
[0038] FIG. **6** illustrates a flow chart of a computer-implemented method for determining an identifier indicative of the location of a computer software using a distributed hash table; and
[0039] FIG. **7** illustrates a schematic of an example processing device.
DESCRIPTION OF EMBODIMENTS
[0040] The present disclosure generally relates to methods and systems for utilising a distributed hash table and a peer-to-peer (P2P) distributed ledger, such as the Bitcoin Blockchain, to enable securing a computer software.
[0041] While embodiments described below may refer specifically to transactions that occur on the Bitcoin Blockchain (referred to herein as the blockchain), it will be appreciated that the present invention may be implemented using other P2P distributed ledgers. The blockchain is used below to describe aspects of the invention for simplicity only due to its high level of standardisation and

large quantity of associated public documentation.

Distributed Hash Table

[0042] In a typical client/server model a central server may be in charge of the majority of resources. This means that in the event of an attack on or failure of the central server, the majority of the resources stored on the central server may be compromised. On the contrary, in a distributed model the resources are shared (“distributed”) between participating nodes. In this way, the capacity of all participating nodes is utilised and the failure of one server will not compromise the majority of the resources.

[0043] FIG. **1** illustrates an example of a hash table. The hash table is comprised of key-value pairs. The key of each key-value pair is mapped, by way of a hash function, to an index. The index defines the location of stored values of the key-value pairs.

[0044] A DHT is an example of applying the distributed model to a hash table. Similar to a hash table, a DHT comprises key-value pairs and provides an efficient method to locate (“lookup”) a value of a key-value pair given just the key. However, in contrast to the hash table, the key-value pairs are distributed and stored by a number of participating nodes. In this way, responsibility for storing and maintaining the key-value pairs is shared by the participating nodes.

[0045] In the same way as a hash table, each key-value pair in the DHT is mapped to an index. The index is determined for each key-value pair by performing a hash function on the key. For example, the cryptographic Secure Hash Algorithm SHA-1 may be used to determine the index.

[0046] Each participating node is assigned at least one index by keyspace partitioning. For each index that the participating node is assigned, the participating node stores the value of that key-value pair.

[0047] It is an advantage that values of the key-value pairs may be efficiently retrieved. To retrieve a value associated with a key, a node may execute a “lookup” to determine the responsible node (via the index). The responsible node may then be accessed to determine the value.

Bitcoin and the Blockchain

[0048] As is well known in the art, a blockchain is a transaction type ledger where storage capacity is distributed across networked nodes participating in a system based on the Bitcoin protocol. Each Bitcoin transaction is broadcast to the network, the transactions are confirmed and then aggregated into blocks. The blocks are then included on the Blockchain by storing the blocks at multiple participating nodes.

[0049] A full copy of a cryptocurrency's P2P distributed ledger contains every transaction ever executed in the cryptocurrency. Thus, a continuously growing list of transactional data records is provided. Since each transaction entered onto the blockchain is cryptographically enforced, the blockchain is hardened against tampering and revision, even by operators of the participating nodes.

[0050] Due to the transparency of the blockchain, histories are publicly available for each transaction.

[0051] It is a further advantage of the blockchain is that the transaction is also the record of the transaction, i.e. the transaction is embedded within the blockchain.

[0052] In this way, the information relating to the transaction is captured in the actual transaction. This record is permanent and immutable and therefore removes the requirement for a third party to keep the transaction record on a separate database. Advantageously, the invention may use techniques to facilitate this control or transfer of an asset, such as software, via a blockchain, and may enable the transfer to be performed in a secure manner, incorporating the use of cryptographic keys, while not requiring any alteration of the underlying blockchain protocol.

Pay-to-Script-Hash and Multi-Signature

[0053] While embodiments below may refer specifically to transactions that use the pay-to-script-hash (P2SH) method of the Bitcoin protocol, it will be appreciated that the present invention may be implemented using another functionally equivalent method of a blockchain Bitcoin protocol.

[0054] Each transaction record on the blockchain comprises a script including information indicative of the transaction and a number of public keys. These public keys may be associated with the sender and recipient of the cryptocurrency. A P2PKH input includes the public key of the sender. A P2PKH output includes the hash of the public key of the recipient. A P2SH multisig input includes the signature of the senders. A script can be considered as a list of instructions recorded with each transaction record on the blockchain that describes how a user may gain access to the cryptocurrency specified in the transaction record.
[0055] As background, in a standard P2SH method of the Bitcoin protocol, the output script, or redeem script, may take the form: [0056] <NumSigs PubK1 PubK2 . . . . PubK15 NumKeys OP_CHECKMULTISIG>
[0057] where NumSigs is the number "m" of valid signatures required to satisfy the redeem script to unlock the transaction; PubK1, PubK2 . . . . PubK15 are the public keys that correspond to signatures that unlock the transaction (up to a maximum of 15 public keys) and NumKeys is the number "n" of public keys.
[0058] In the Bitcoin protocol, signatures based on a user's private key may be generated using the Elliptic Curve Digital Signature Algorithm. The signatures are then used for redemption of the cryptocurrency associated with the output script or redeem script. When a user redeems an output script or redeem script, the user provides their signature and public key. The output script or redeem script then verifies the signature against the public key.
[0059] To redeem the above redeem script, at least a number "m" of signatures corresponding to the public keys are required. In some examples, the order of the public keys is important and the number "m" out of "n" signatures for signing must be done in sequence. For example, consider where "m" is 2 and "n" is 15. If there are two signatures available for use, Sig1 (corresponding to PubK1) and Sig 15 (corresponding to PubK15), the redeem script must be signed by Sig1 first followed by Sig15.
Overview of the System
[0060] A method, device and system for determining a metadata (M) for securing computer software will now be described.
[0061] FIG. **2** illustrates a system **1** that includes a first node **3** that is in communication with, over a communications network **5**, a second node **7**. The first node **3** has an associated first processing device **21** and the second node **5** has an associated second processing device **27**. Examples of the first and second nodes **3**, **7** include an electronic device, such as a computer, tablet computer, mobile communication device, computer server etc.
[0062] A DHT **13** to record and store key-value pairs is also illustrated in FIG. **2**. The DHT **13** may be associated with one or more processing devices **19** to receive, record and store the values of the key-value pairs. The processing devices **19** may be used by "participating nodes" of the DHT **13**. As described above, the DHT **13** provides an efficient method to locate values of key-value pairs.
[0063] FIG. **2** also illustrates a P2P distributed ledger **14** to record transactions. The P2P distributed ledger **14** may be associated with one or more processing devices **20** to receive and record transactions. As described above, an example of a P2P distributed ledger **14** is the Bitcoin Blockchain. Therefore, in the context of the blockchain, the processing devices **20** associated with the P2P distributed ledger **14** may be processing devices referred to as "miners".
[0064] The first node **3** is associated with a first user **23** and the second node **7** is associated with a second user **24**. In one example, the first node **3** may represent a vendor of the computer software. In another example, the first node **3** may represent an agent or service provider. In yet another example, the first node **3** may represent a user of the computer software.
[0065] Similarly, the second node **7** may represent the agent, service provider, vendor of the computer software or a user of the computer software.
[0066] In one example, the first node **3** performs the method **100**, **500** as illustrated by FIG. **3** and FIG. **6**. In another example, the second node **7** performs the method **100**, **500**. While the exemplary

embodiments below may refer to the first node **3** as performing the methods it is to be understood the disclosure may also be adapted or modified to be performed by other nodes.
[0067] The method **100** as illustrated by FIG. **3** includes determining **110** a data (D1) associated with the computer software. The data (D1) may further comprise a licence associated with the computer software. The method **100** also includes determining **120** a first hash value (H1) based on the computer software. In one example, the first hash value (H1) may be in relation to an executable of the computer software.
[0068] The method **100** also includes determining **130** a second hash value (H2) based on the data (D1) and the computer software. In one example, the second hash value (H2) may be representative of the details of the computer software and the licence associated with the computer software. In a further example, the second hash value (H2) may comprise additional information.
[0069] The method **100** further includes sending **140**, over a communications network **5**, the data (D1), the first hash value (H1) and the second hash value (H2) to an entry on a distributed hash table 13, wherein the second hash value (H2) is assigned to a key of a key-value pair and the data (D1) and the first hash value (H1) are assigned to the value in the key-value pair. The value in the key-value pair may further comprise an identifier indicative of the location of the computer software or licence.
[0070] The method **100** also includes determining **150** a metadata (M) that is based on the second hash value (H2) for inclusion on the peer-to-peer distributed ledger **14**. In one example, the metadata (M) may be included in a first redeem script (RS1) for inclusion on the peer-to-peer distributed ledger **14**.
[0071] A detailed example of the method will now be described.
Determining a Data Associated with the Computer Software **110**
[0072] As described above the method **100** includes determining **110** a data (D1) associated with the computer software. Determining **110** a data (D1) may comprise receiving the data (D1) from a user, node or data store. Determining **110** a data (D1) may further comprise generating the data (D1) at the first node **3**.
[0073] In one example, the first node **3** may receive the data (D1) from the first user **23** via the user interface **15**. In another example, the first node **3** may receive the data (D1) from the second user **24**. In yet another example, the first node **3** may receive the data (D1) from a data store **17**.
[0074] Data (D1) is associated with the computer software where data (D1) may identify the computer software, additional information, a licence of the computer software or be indicative of the location of the computer software. For example, the data (D1) may comprise a string or data structure that identifies the computer software. The string or data structure may comprise a collection of identifying keywords and/or additional information about the computer software. An example of additional information may be an identifier of the version of the computer software, for example a numeral. For instance, if the computer software is entitled BobSoftware and the version is 3.0, the string or data structure (D1) may comprise “BobSoftware/3.0”.
[0075] In a further example, the data (D1) may comprise an identifier of a licence associated with the computer software. This may be a software licence identification number (ID) or a software licence key. In another example, the identifier of the licence may comprise a cryptographic hash of the contents of the licence.
[0076] The data (D1) may further comprise an identifier indicative of the storage location of the computer software. In one example, the identifier may comprise a URL for an object on the Internet. In a further example, a link to the storage location of the computer software on a repository such as a hash table or distributed hash table may be provided.
[0077] In yet a further example the data (D1) may comprise information that identifies the vendor of the computer software. This may include personal details such as name, address, contact details or a public key associated with the vendor.
Determining a First Hash Value (H1) Based on the Computer Software **120**

[0078] As also described above the method **100** further includes determining **120** a first hash value (H1) of the computer software. Determining **120** a first hash value (H1) may comprise receiving the first hash value (H1) from a user or accessing the first hash value (H1) from a data store. Determining **120** a first hash value (H1) may further comprise calculating the hash value at the first node **3**.
[0079] In one example, the first node **3** may receive the first hash value (H1) from the first user **23** via the user interface **15**. In another example, the first node **3** may receive the first hash value (H1) from the second user **24**. In yet another example, the first node **3** may access the first hash value (H1) from a local data store **17** or remote data store.
[0080] In one example, the first hash value (H1) is of an executable of the computer software. The executable of the computer software may be retrieved from the communications network **5** such as the Internet. In another example, the executable may be provided by the first user **23** or the second user **24**. In yet another example, the executable may be retrieved from the data store **17**. In yet a further example, the executable may be retrievable from a repository such as a hash table or a DHT.
[0081] The hash of the executable of the software may be determined using the SHA-256 algorithm to create a 256-bit representation of the information. It is to be appreciated that other hash algorithms may be used, including other algorithms in the Secure Hash Algorithm (SHA) family. Some particular examples include instances in the SHA-3 subset, including SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, SHAKE256. Other hash algorithms may include those in the RACE Integrity Primitives Evaluation Message Digest (RIPEMD) family. A particular example may include RIPEMD-160. Other hash functions may include families based on the Zémor-Tillich hash function and knapsack-based hash functions.
Determining a Second Hash Value (H2) Based on the Data (D1) and the Computer Software **130**
[0082] The method **100** also includes determining **130** a second hash value (H2) based on the data (D1) and the computer software.
[0083] In one example, the second hash value (H2) may be determined based on the hash of the concatenation of the data (D1) and the executable (or hash of the executable, that is, the first hash value (H1)) of the computer software. In a further example, the second hash value (H2) may be determined based on the hash of the concatenation of the data (D1), the executable (or hash of the executable) of the computer software and additional information.
[0084] Additional information may comprise a public key of the first user **23** (PU1) or second user **24** (PU2). In a further example the additional information may comprise an identifier of an entity associated with the first user **23** or second user **24**. For instance, the entity may be an employer of the first user **23** or second user **24**. In another example, the entity may be a service provider of the first user **23** or second user **24**.
[0085] The additional information may further comprise a device identifier of a device associated with the first node **3**, second node **7**, first user **23** or second user **24**. An example of a device is the first processing device **21** as illustrated in FIG. **2**. The device identifier may comprise at least one of the following: a MAC address, motherboard serial number or a device identification number. The device identifier may further be a concatenation of at least two of the MAC address, motherboard serial number or device identification number. In a further example the device identifier may comprise a hash value associated with the MAC address, motherboard serial number or device identification number, or the concatenation described above.
[0086] In yet a further example the additional information may comprise an expiry date of the licence associated with the computer software.
Licence Associated with the Computer Software
[0087] In a further example, the second hash value (H2) may be determined based on the concatenation of the data (D1), the executable (or hash of the executable) of the computer software, additional information or the licence that relates to the computer software.
[0088] The representation of the licence may be a file or document which specifies the content of

the licence. For example, plain ASCII text, a PDF document or a Word document. The second hash value (H2) may include the licence in its original form, or for example it may provide a link to the location of the licence on a publicly accessible communications network such as the Internet. In a further example, a link to the location of the licence on a repository such as a hash table or DHT may be provided. In yet a further example, a link to the location of the licence on a computer-based resource, such as the data store **17** may be provided.
[0089] In one example, the licence may comprise the first hash value (H1) associated with the computer software.
[0090] The licence associated with the computer software may further comprise additional information as described above. In one example, the licence may be associated with a first user **23** or second user **24**. The licence may comprise the public key of the first user **23** (PU1) or second user **24** (PU2). In a further example the licence may comprise an identifier of an entity associated with the first user **23** or second user **24**.
[0091] The licence associated with the computer software may further comprise a device identifier of a device associated with the first node **3**, second node **7**, first user **23** or second user **24**. An example of a device is the first processing device **21** as illustrated in FIG. **2**. The device identifier may comprise at least one of the following: a MAC address, motherboard serial number or a device identification number. The device identifier may further be a concatenation of at least two of the MAC address, motherboard serial number or device identification number. In a further example the device identifier may comprise a hash value associated with the MAC address, motherboard serial number or device identification number, or the concatenation described above.
[0092] The first user **23** may be the vendor of the computer software and the second user **24** may be the recipient ("end user") of the computer software. In another example the second user **23** may be the vendor of the computer software and the second user **24** may be the end user of the computer software.
[0093] In one example the licence associated with the computer software may authorise only one end user (a "single-user licence"). In a further example, the licence associated with the computer software may authorise one device of the end user (a "single-device licence"). In another example the licence associated with the computer software may authorise more than one device of the end user (a "multi-device licence").
[0094] In another example, there may be more than one end user (a "multi-user licence"). In a further example, the licence associated with the computer software may authorise one device per end user. In another example the licence associated with the computer software may authorise more than one device per end user.
[0095] In the event that the licence is associated with a first user **23** or a second user **24**, the licence may comprise the first user public key (PU1) associated with the first user **23** and the second user public key (PU2) associated with the second user **24**.
Merkle Tree
[0096] In another example, the licence may be the top hash value of a Merkle tree. An example of a Merkle tree is illustrated in FIG. **4**. In a Merkle tree, the hash value at each node are hashes of their respective "child" nodes. For example, the hash value Hash-A **305** is the hash of the hash values at the two "child" nodes **309** and **311**. It can be seen that the top hash value of the Merkle tree, Hash-AB **303**, comprises all the hash values in the Merkle tree. That is, it captures the hash values of the four "leaves" at the bottom of the tree, A1 **317**, A2 **319**, B1 **321** and B2 **323**.
[0097] In an example of the present disclosure, each "leaf" of the Merkle tree may represent an aspect of the information of the licence. An exemplary licence is illustrated in FIG. **5**. The data (D1) **417** is captured in the hash value Hash-D **409**, the executable of the software **419** is captured in the hash value Hash-S **411** (H1), the public keys **421** of users **23** and/or **24** are captured in the hash value Hash-P **413** and the expiry date **423** is captured in the hash value Hash-E **415**. It can be seen that the nodes **405** and **407** capture the hash values associated with the leaves for data (D1)

417 and software **419**, and public keys **421** and expiry date **423** respectively.
[0098] It is to be appreciated that other information not otherwise described above may comprise the additional information that the hash value (H2) is based on.
Sending the Data (D1), First Hash Value (H1) and Second Hash Value (H2) to a Distributed Hash Table **140**
[0099] The method **100** also includes sending **140**, over a communications network **5**, the data (D1), first hash value (H1) and the second hash value (H2) to an entry on a distributed hash table 13.
[0100] In one example, the second hash value (H2) may be a key of a key-value pair, and the data (D1) and the first hash value (H1) may be a value in the key-value pair.
[0101] In a further example, additional information as described above may also be part of the value in the key-value pair. This includes but is not limited to: public keys of the first user **23** or second user **24**, a device identifier of a device associated with the first node **3**, second node **7**, first user **23** or second user **24**, an identifier indicative of the location of the computer software or licence, or further additional information associated with the licence.
[0102] As described above, a DHT **13** is comprised of key-value pairs, where each key-value pair is assigned to an index. In one example, the second hash value (H2) may be used to generate the index. A hash function or cryptographic hash function may be performed on the second hash value (H2). For instance, the cryptographic function SHA-1 may be used:
[00001]$\text{Index} = \text{SHA-1}(H2)$
[0103] For the second hash value (H2) to be the key of a key-value pair in the DHT **13**, and the data (D1) and the first hash value (H1) to be the value in the key-value pair, the key and value are sent to any participating node of the DHT **13**.
[0104] In one example, a message such as put (key, value) may be sent to a participating node of the DHT **13**, where key is the second hash value (H2) and value is the data (D1) and the first hash value (H1). The message may be sent around all participating nodes until it is received by the participating node that is assigned to the index as indicated by the keyspace partitioning. The participating node assigned to the index indicated in the message may then store the key-value pair on the DHT **13** and assume responsibility for maintaining the entry associated with the key-value pair.
[0105] It is an advantage that the value of any given key may be retrieved from the DHT **13**. In one example, the first user **23** or second user **24** may wish to retrieve the value. The first user **23** or second user **24**, via the first node **3**, second node **7** or another node not otherwise illustrated, may provide any participating node of the DHT **13** a request message such as get (key). The request message may then be sent around all participating nodes until it is received by the participating node that is assigned to the index as indicated by the keyspace partitioning.
Determining a Metadata (M) **150**
[0106] The method **100** further includes determining **150** a metadata (M) that comprises the second hash value (H2). Determining **150** a metadata (M) may comprise receiving the metadata (M) from a user, node or data store. The metadata (M) may be included in, for example, in one or more of the 15 places available for the public keys in a P2SH multi-signature first redeem script (RS1) of a transaction on the P2P distributed ledger (blockchain) **14**.
[0107] The first redeem script (RS1) of the transaction on the P2P distributed ledger **14** may represent an issuance, or creation, of a tokenised transaction (“issuance token”) that represents the content included in the metadata (M). In one example, the token may be issued by an agent (A).
[0108] In the P2SH method of the Bitcoin protocol, metadata may be included in a redeem script by way of the method described below.
Metadata
[0109] Metadata (M) may be embedded in one or more of the 15 places available for the public keys in a P2SH multi-signature redeem script (RS1). For example, the redeem script (RS1) may

take the form of: [0110] <NumSigs Metadata1 Metadata2 . . . . PubK1 PubK2 . . . . NumKeys OP_CHECKMULTISIG>
where Metadata1 and Metadata2 each include metadata that takes the place of a public key in the redeem script and PubK1 and PubK2 are public keys. In other words, the metadata may be provided in the redeem script at a location which is designated by the blockchain protocol as a position where a cryptographic key should be provided. This provides the advantage that the metadata can be incorporated into the transaction (Tx) without any change to the underlying blockchain protocol.
[0111] Metadata (M) may comprise the second hash value (H2). The metadata (M) may further comprise a description or keyword describing conditions associated with the computer software or licence. For example, the date of the licence, name, date of birth, address, contact details, or other details of the user associated with the licence. In a further example, information associated with the quantity of cryptocurrency may be included.
[0112] The metadata (M) may include the information in a number of ways. In one example, the contents of the information may be included. In a further example, a cryptographic hash of the information may be included. The hash of the information may be determined using the SHA-256 algorithm to create a 256-bit representation of the information. It is to be appreciated that other hash algorithms may be used, including other algorithms in the Secure Hash Algorithm (SHA) family. Some particular examples include instances in the SHA-3 subset, including SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, SHAKE256. Other hash algorithms may include those in the RACE Integrity Primitives Evaluation Message Digest (RIPEMD) family. A particular example may include RIPEMD-160. Other hash functions may include families based on Zémor-Tillich hash function and knapsack-based hash functions.
[0113] In further embodiments of the present disclosure, combinations including one or more of the above may be included in the metadata (M). Since the metadata (M) may be made public by way of the P2P distributed ledger **14** such as the blockchain, or transmitted over an unsecure network, it may be desirable that specific details of the metadata (M) be veiled or hidden for privacy reasons.
[0114] Therefore, the use of multi-signature P2SH Bitcoin transactions in embodiments of the present disclosure offers an advantage as it enables the transfer and permanent record of information associated with the computer software and the licence. This record is achieved by including the metadata in the output script of a transaction, for example, a redeem script.
First Redeem Script
[0115] As described above, a redeem script is an example of an output script in the standard P2SH method of the Bitcoin protocol and describes how a user may gain access to the cryptocurrency specified in the transaction record.
[0116] In the present disclosure the first redeem script (RS1) for the issuance token may be based on the metadata (M). The first redeem script (RS1) may further comprise an agent public key (PA) that forms a cryptographic pair with an agent private key (VA). In this way, the agent private key (VA) is required to "unlock" or spend cryptocurrency that is associated with the transaction.
[0117] In one example, the first redeem script (RS1) for the issuance token may include the metadata (M). The first redeem script (RS1) may further comprise an agent public key (PA). In this example, the first redeem script (RS1) may be of the form: [0118] <OP_1 PA Metadata1 Metadata2 OP_3 OP_CHECKMULTISIG>
where OP_1 denotes the number of signatures required to satisfy the first redeem script (RS1) to unlock the transaction ("NumSigs"), and OP_3 denotes the number of public keys in the redeem script ("NumKeys").
[0119] In this example, the first redeem script (RS1) may comprise two designated fields for the metadata, Metadata1 and Metadata2. A specific example of the Metadata1 and Metadata2 is illustrated in Table 1 below.
TABLE-US-00001 TABLE 1 Field Sub-field Bytes Comments Metadata1 LicenceType 4 Coded

value indicates type of licence. LicencePointer 16 IPv6 address identifying the DHT. LicenceTypeData1 12 Format depends on value of LicenceType. Padded with zeros. Metadata2 LicenceHash 20 RIPEMD- 160(SHA256(actual licence file addressed by LicencePointer)) LicenceTypeData2 12 Format depends on value of LicenceType. Padded with zeros.
[0120] This example includes providing a pointer to the licence in Metadata1 which may be useful where the size of the licence precludes including such details in the metadata (M). Furthermore, since the metadata (M) may be made public, or transmitted over an unsecure network, it may be desirable that specific details of the token be veiled or hidden for privacy reasons.
[0121] The first 4 bytes of Metadata1 indicates the type of licence. For example, the licence type may denote the name of the computer software such as BobSoftware. In a further example the licence type may denote the authorisation type of the licence, such as “single-user” or “multi-device” as described above. The next 16 bytes holds the IP address of the location of the actual electronic licence file, making allowance for IPV6 addresses. Note that in some embodiments, this value may point to the seed of a torrent file such that the licence file can be distributed over the cloud rather than being centralised. The following 12 bytes contains data specific to the type of licence.
[0122] The first 20 bytes of Metadata2 is a hash of the actual licence file using RIPEMD-160 over SHA256 applied to the actual contents of the licence file. As the actual licence file may be retrievable this allows validation of the transaction against the contract. Note that the licence file itself may be completely public (unencrypted and human readable) or may be encrypted for privacy, depending on the requirements of the specific embodiment. The content of the remaining 12 bytes of Metadata2 may be used depending on the type of licence.
[0123] It can be seen from the example of the first redeem script (RS1) provided above that the issuance token must be signed by the agent (A) in order to be spent. An example of the transaction for the issuance token is provided in Table 2, where for brevity the miner's fee is not shown. TABLE-US-00002 TABLE 2 ID-600 Transaction-ID Version number Version number 1 Number of inputs ID-110 Prev Trans Output IDX-00 Prev Trans Output index Script length Script length OP_0 Sig-VA < redeem script ID-110 > ScriptSig 0000 0000 0000 0001 Sequence number 1 Number of outputs C1 Output value Output script length Output script length OP_HASH160 < hash of redeem script (RS1) > Output script OP_EQUAL LockTime LockTime
[0124] Lines 4 to 8 of Table 2 represent the input to the transaction which is the first quantity of cryptocurrency (C1) that is to be included in the issuance token (i.e. “tokenised”). In this example, the first quantity of cryptocurrency (C1) was the result of a previous transaction (ID-110) that transferred the first quantity of cryptocurrency to the benefit of the agent (A), and therefore the previous transaction (ID-110) output script (redeem script ID-110) includes the agent's public key (PA). Accordingly, to unlock this previous output the script (redeem script ID-110) must be signed with the first user's private key (VA).
[0125] Lines 10 to 12 of Table 2 represent the first (and only) output of the transaction (ID-600), which in this case is representative of the issuance token being created and transferred back to the agent. Line 10 shows the output value, which is the first quantity of cryptocurrency (C1). Line 12 shows the output script, which includes a “<hash of redeem script >” as is used in the P2SH method of the Bitcoin protocol. In this example, the redeem script is the first redeem script (RS1) in the form as described above.
[0126] The output of the transaction (ID-600) shown in Table 2 is then recorded, with the first data output (O1), on the P2P distributed ledger **14**. In particular, the first data output (O1) may comprise an indication of the first quantity of cryptocurrency (C1) that was transferred in the transaction. The first data output (O1) may further comprise a hash of the first redeem script (RS1).
[0127] In future transactions of the first quantity of cryptocurrency (C1), for example the transfer of the token to a first user **23** or second user **24**, the script to unlock the first quantity of cryptocurrency (C1) (e.g. the input ScriptSig of the future transaction) may be in the form: [0128]

OP_0 Sig-VA Sig-VU1<OP_1 PA PU1 Metadata1 Metadata2 OP_4 OP_CHECKMULTISIG> where Sig-VU1 indicates the signature of the first user **23**. Note that the above script assumes that only one signature from the agent (A) or the first user **23** is required to unlock the first quantity of cryptocurrency (C1).
[0129] The issuance token may be transferred to another user by way of a second redeem script (RS2).
VARIATIONS
Second Redeem Script
[0130] The token that is associated with the computer software and licence may be transferred from the agent (A) to another user, for example the first user **23** or second user **24**. In one example, the transfer of the token may be representative as authorising access to the user for the computer software or licence. The transfer may be implemented by a second redeem script (RS2).
[0131] In one example, the agent (A) wishes to transfer the issuance token to a first user **23**. The first user **23** may represent, for example, a vendor of the computer software.
[0132] In this example, the second redeem script (RS2) may be based on the metadata (M), the agent public key (PA) associated with the agent (A) and the first user public key (PU1) associated with the first user **23**.
[0133] The second redeem script (RS2) may be of the form: [0134] <OP_1 PA PU1 Metadata1 Metadata2 OP_4 OP_CHECKMULTISIG>
[0135] In this example, the second redeem script (RS2) comprises the same two metadata fields as the first redeem script (RS1). The second redeem script (RS2) further comprises the agent public key (PA) associated with the agent and the first user public key (PU1) associated with the first user.
[0136] It can be seen from the example of the second redeem script (RS2) provided above that the token that is transferred must be signed by the agent (A) or the first user **23** in order to be spent. An example of the transaction for this transfer of the issuance token is provided in Table 3, where again for brevity the miner's fee is not shown.
TABLE-US-00003 TABLE 3 ID-610 Transaction-ID Version number Version number 1 Number of inputs ID-600 Prev Trans Output IDX-00 Prev Trans Output index Script length Script length OP_0 Sig-VA < OP_1 PA Metadata1 ScriptSig Metadata2 OP_3 OP_CHECKMULTISIG > 0000 0000 0000 0001 Sequence number 1 Number of outputs C1 Output value Output script length Output script length OP_HASH160 < hash of redeem script (RS2) > Output script OP_EQUAL LockTime LockTime
[0137] Similar to Table 2, lines 4 to 8 of Table 3 represent the input to the transaction (ID-610). In this example, the input is the issuance token, i.e. the output of the transaction (ID-600) that is illustrated in Table 2. It can be seen that the redeem script in line 7 corresponds to the redeem script of the issuance token, i.e. the first redeem script (RS1). Accordingly, to unlock the output of the transaction (ID-600) the first redeem script (RS1) must be signed with the agent's public key (PA).
[0138] Lines 10 to 12 of Table 3 represent the output of the transaction (ID-610), which in this case is representative of the issuance token being transferred to either the agent (A) or the first user **23** (U1). Line 10 shows the output value, which is the first quantity of cryptocurrency (C1). Line 12 shows the output script, which includes a "<hash of redeem script >" as is used in the P2SH method of the Bitcoin protocol. In this example, the redeem script is the second redeem script (RS2) in the form as described above.
[0139] The output of the transaction (ID-610) is then recorded, with a second data output (O2), on the P2P distributed ledger **14**. The second data output (O2) may comprise an indication that the first quantity of cryptocurrency (C1) from the first data output (O1) is to be transferred in the transaction. The second data output (O2) may further comprise a hash of the second redeem script (RS2).
Identifier Indicative of the Location of the Computer Software or Licence
[0140] As described above the data (D1) or licence may comprise an identifier indicative of the

location of the computer software or licence respectively.
[0141] In one example, the identifier may be determined independently to the data (D1) or the licence and remain separate to the data (D1) or licence. The identifier may further be assigned to the value of the key-value pair together with the data (D1) and the first hash value (H1) as described in the method **100** above. In this way, the identifier may be included in the value field of the message put (key, value) and sent to a participating node in the DHT **13**, as described above.
[0142] In one example, the identifier indicative of the location may comprise a URL for an object on the Internet. In another example, the identifier indicative of the location may comprise an address for a repository such as a hash table or a DHT **13**. In yet another example, the identifier indicative of the location may comprise an address for a computer-based repository such as a server, database or storage facility provided on a computer-based resource, such as the data store **17** associated with the first processing device **21** of the first node **3**.
[0143] FIG. **6** illustrates a method **500** for determining location of the computer software or licence. The method **500** includes determining **510** the metadata (M) from the first redeem script (RS1). As described above, the metadata (M) may be embedded in one or more of the 15 places available for the public keys in the first redeem script (RS1).
[0144] In the P2SH method of the Bitcoin protocol, when the output of a transaction is spent in a subsequent transaction, the redeem script becomes visible in the subsequent transaction. As described above and with reference to Table 2, the transaction (ID-600) for the issuance token is paid back to the agent (A). In this way, the agent (A) may spend this issuance token to expose the first redeem script (RS1). The metadata (M) that is based on the second hash value (H2) is therefore visible on the P2P distributed ledger **14**. In this way, the second hash value (H2) is able to be retrieved **520** from the metadata (M) in the first redeem script (RS1). In one example, the value associated with the key of the key-value pair is able to be retrieved from the DHT **13** using the request message get (key).
[0145] The method **500** further includes sending **530**, over a communications network **5**, the second hash value (H2) to a processor associated with a participating node of the DHT **13**. As described above, the second hash value (H2) may be the key of the key-value pair. As also described above, the value for a given key may be retrieved by providing a message containing the key to any participating node of the DHT **13**. Therefore, in the example where the identifier is included in the value field of the key-value pair, the method **500** is able to determine **540**, from the processor of the participating node, the identifier indicative of the location of the computer software or licence.
Processing Device
[0146] As noted above, the first 3 and second node **7** may be an electronic device, such as a computer, tablet computer, mobile communication device, computer server etc. The electronic device may include a processing device **21**, **27**, a data store **17** and a user interface **15**.
[0147] FIG. **7** illustrates an example of a processing device **21**, **27**. The processing device **21**, **27** may be used at the first node **3**, second node **7** or other nodes not otherwise illustrated. The processing device **21**, **27** includes a processor **1510**, a memory **1520** and an interface device **1540** that communicate with each other via a bus **1530**. The memory **1520** stores a computer software program comprising machine-readable instructions and data for implementing the method **100** and **500** described above, and the processor **1510** performs the instructions from the memory **1520** to implement the method **100** and **500**. The interface device **1540** may include a communications module that facilitates communication with the communications network **5**, and in some examples, with the user interface **15** and peripherals such as data store **17**. It should be noted that although the processing device **1510** may be an independent network element, the processing device **1510** may also be part of another network element. Further, some functions performed by the processing device **1510** may be distributed between multiple network elements. For example, the first node **3** may have multiple processing devices **21** to perform method **100**, **500** in a secure local area

network associated with the first node **3**.
[0148] Where this disclosure describes that a user, employer, employee, issuer, merchant, provider or other entity performs a particular action (including signing, issuing, determining, calculating, sending, receiving, creating etc.), this wording is used for the sake of clarity of presentation. It should be understood that these actions are performed by the computing devices operated by these entities.
[0149] It will be appreciated by persons skilled in the art that numerous variations and/or modifications may be made to the above-described embodiments, without departing from the broad general scope of the present disclosure. The present embodiments are, therefore, to be considered in all respects as illustrative and not restrictive.

Claims

1. A computer-implemented method for securing a controlled digital resource using a distributed hash table and a peer-to-peer distributed ledger, the method comprising: determining a data (D1) associated with the controlled digital resource; determining a first hash value (H1) of the controlled digital resource; determining a second hash value (H2) based on the data (D1) and the controlled digital resource; sending, over a communications network, the data (D1), the first hash value (H1) and the second hash value (H2) to an entry for storage in a distributed hash table, wherein the second hash value (H2) is a key of a key-value pair and the data (D1) and the first hash value (H1) are a value in the key-value pair; and determining a metadata (M) comprising the second hash value (H2) for storage on the peer-to-peer distributed ledger.