METHODS AND DEVICES FOR VERIFYING ATTRIBUTE VALUE OF ASSET REPRESENTED BY TOKEN RECORDED ON BLOCKCHAIN

TECHNICAL FIELD

[0001] The specification relates generally to computer technologies, and more particularly, to methods and devices for verifying an attribute value of an asset represented by a token recorded on a blockchain.

BACKGROUND

[0002] Conventionally, a fungible asset is generally composed of units that can be interchanged, and a non-fungible asset may not be interchanged because it has unique properties. Both types of assets may exist in the digital world, and may be bought and sold like their physical counterparts.

[0003] A non-fungible token (NFT) may be a unit of data utilized to certify that an asset is unique and not interchangeable (i.e., the asset is non-fungible). NFTs may be utilized to certify items such as artworks, photos, videos, audio files, transactions, records, and other types of non-fungible assets. For example, while traditional works of art such as paintings and the like are one of a kind, digital artworks can be easily and repeatedly duplicated. Using NFTs, a digital artwork (or other types of assets) may be “tokenized” to create a digital certificate of ownership to certify that the digital artwork is unique and not interchangeable.

[0004] NFTs may be bought and sold, and may be stored on a digital ledger, which may be implemented using a blockchain system. Blockchain systems, also known as distributed ledger systems (DLSs) or consensus systems, may enable participating parties to store data securely and immutably. Blockchain systems may

include any DLSs, without referencing any particular use case, and may be used for public, private, and consortium blockchain networks. A public blockchain network is open for all entities to use the system and participate in the consensus process. A private blockchain network is provided for a particular entity, which centrally controls read and write permissions. A consortium blockchain network is provided for a select group of entities, which control the consensus process, and includes an access control layer.

[0005] A blockchain system is implemented using a peer-to-peer (P2P) network, in which the nodes communicate directly with each other, e.g., without the need of a fixed, central server. Each node in the P2P network may initiate communication with another node in the P2P network. A blockchain system maintains one or more blockchains. A blockchain is a data structure for storing data, such as transactions, that may prevent tampering and manipulation of the data by malicious parties.

[0006] Users of a blockchain system may utilize the blockchain system to record various types of information, including, e.g., NFTs. Currently, many NFT applications have been made available on various blockchain platforms, including, e.g., Ethereum, Binance Smart Chain, and the like. Many types of NFTs have been minted and traded on these blockchain platforms, including, e.g., artworks, music, domain names, trading cards, collectibles, and the like. Other types of assets, including invoices and purchase orders, may also be represented using non-fungible tokens and traded on such blockchain platforms. However, little work has been done with respect to protecting the privacy of the NFTs recorded on blockchains.

SUMMARY

[0007] In one aspect, a computer-implemented method for generating a privacy-preserving token includes calculating a privacy-preserving representation of a set of attributes associated with an asset, wherein calculating the privacy-preserving representation comprises calculating a Merkle tree root hash based on a value associated with each attribute of the set of attributes, wherein the value associated with each attribute of the set of attributes has a corresponding hash stored on a leaf node of the Merkle tree; and recording the privacy-preserving representation of the set of attributes on a blockchain system as metadata of a token representing the asset.

[0008] In another aspect, a computer-implemented method for verifying an attribute value of an asset represented by a privacy-preserving token, wherein a privacy-preserving representation of a set of attributes associated with the asset is calculated by calculating a Merkle tree root hash based on a value associated with each attribute of the set of attributes and is recorded on a blockchain system as metadata of the token, and the value associated with each attribute of the set of attributes has a corresponding hash stored on a leaf node of the Merkle tree is disclosed. The method includes: generating a proof for proving that the value associated with a particular attribute of the set of attributes satisfies a condition; and providing the proof to a verifier to facilitate a verification of whether the value associated with the particular attribute of the set of attributes satisfies the condition without revealing the value associated with the particular attribute.

[0009] In still another aspect, a computer-implemented method for verifying an attribute value of an asset represented by a privacy-preserving token, wherein a privacy-preserving representation of a set of attributes associated with the asset is calculated by calculating a Merkle tree root hash based on a value associated with

each attribute of the set of attributes and is recorded on a blockchain system as metadata of the token, and the value associated with each attribute of the set of attributes has a corresponding hash stored on a leaf node of the Merkle tree is disclosed. The method includes: receiving a proof for proving that the value associated with a particular attribute of the set of attributes satisfies a condition; and verifying whether the value associated with the particular attribute of the set of attributes satisfies the condition without knowing the value associated with the particular attribute.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate embodiments. In the following description, which refers to the drawings, the same numbers in different drawings represent the same or similar elements unless otherwise represented.

[0011] FIG. 1 is a schematic diagram of a blockchain system, according to an embodiment.

[0012] FIG. 2 is a schematic diagram of a computing device for implementing a node in a blockchain system, according to an embodiment.

[0013] FIG. 3 is a schematic diagram depicting an implementation of a Merkle tree, according to an embodiment.

[0014] FIG. 4 is a schematic diagram depicting a Merkle tree with hash values stored therein, according to an embodiment.

[0015] FIG. 5 is a schematic diagram depicting a Merkle tree with hash values stored therein, according to an embodiment.

[0016] FIG. 6 is a schematic diagram depicting a Merkle tree with hash values stored therein, according to an embodiment.

[0017] FIG. 7 is a schematic diagram depicting a Merkle tree with hash values stored therein, according to an embodiment.

[0018] FIG. 8 is a flow chart of a method for verifying an attribute value of an asset, according to an embodiment.

[0019] FIG. 9 is a flow chart of a method for verifying an attribute value of an asset, according to an embodiment.

[0020] FIG. 10 is a block diagram of an apparatus for verifying an attribute value of an asset, according to an embodiment.

DETAILED DESCRIPTION

[0021] Embodiments of the specification provide methods and devices for verifying attribute values of assets represented by tokens recorded on blockchains. The methods and devices may calculate a privacy-preserving representation of a set of attributes associated with an asset. The methods and devices may record the privacy-preserving representation on a blockchain system as metadata of a token representing the asset. The methods and devices may also generate a proof for proving that the value associated with a particular attribute of the set of attributes satisfies a condition. The methods and devices may further provide the proof to a verifier to facilitate a verification of whether the value associated with the particular attribute of the set of attributes satisfies the condition without revealing the value associated with the particular attribute.

[0022] Embodiments disclosed in the specification have one or more technical effects. In some embodiments, the methods and devices may calculate the privacy-

preserving representation by calculating a Merkle tree root hash based on values associated with a set of attributes associated with an asset, where the value associated with each attribute of the set of attributes may have a corresponding hash stored on a leaf node of the Merkle tree. This allows the methods and devices to account for each attribute of the set of attributes when calculating the privacy-preserving representation. In some embodiments, the methods and devices may record the privacy-preserving representation on a blockchain system as metadata of a token representing the asset. This allows the methods and devices to record the metadata of the token securely and immutably without exposing any private information about the token on the blockchain. In some embodiments, the methods and devices may generate a proof for proving that the value associated with a particular attribute of the set of attributes satisfies a condition. This allows the methods and devices to provide a mechanism that not only protects the privacy of the tokens, but also safeguards the correctness of the tokens recorded on the blockchain system. In some embodiments, the methods and devices may provide the proof to a verifier to facilitate a verification of whether the value associated with the particular attribute of the set of attributes satisfies the condition without revealing the value associated with the particular attribute. This allows the methods and devices to facilitate the verification of the token while still preserving the privacy of the token being verified.

[0023] A blockchain is a data structure that stores data, e.g., transactions, in a way that may prevent tampering and manipulation of the data by malicious parties. The transactions stored in this manner may be immutable and subsequently verified. A blockchain includes one or more blocks. Each block is linked to a previous block immediately before it in the blockchain by including a cryptographic hash of the previous block. Each block also may include a timestamp, its own cryptographic hash,

and one or more transactions. The transactions, which generally have already been verified by the nodes of the blockchain system, may be hashed and encoded into a data structure, such as a Merkle tree. In a Merkle tree, data at leaf nodes of the tree is hashed, and all hashes in each branch of the tree may be concatenated at a root of the branch. This process continues up the tree to the root of the entire tree, which stores a hash that is representative of all data in the tree. A hash purporting to be of a transaction stored in the tree can be quickly verified by determining whether it is consistent with the structure of the tree.

[0024] A blockchain system includes a network of computing nodes that manage, update, and maintain one or more blockchains. The network may be a public blockchain network, a private blockchain network, or a consortium blockchain network. For example, numerous entities, such as hundreds, thousands, or even millions of entities, can operate in a public blockchain network, and each of the entities operates at least one node in the public blockchain network. Accordingly, the public blockchain network can be considered a public network with respect to the participating entities. Sometimes, a majority of entities (nodes) must sign every block for the block to be valid and added to the blockchain of the blockchain network. Examples of public blockchain networks include particular peer-to-peer payment networks that leverage a distributed ledger, referred to as blockchain.

[0025] In general, a public blockchain network may support public transactions. A public transaction is shared with all of the nodes in the public blockchain network, and is stored in a global blockchain. A global blockchain is a blockchain replicated across all nodes, and all nodes are in perfect state consensus with respect to the global blockchain. To achieve consensus (e.g., agreement to the addition of a block to a blockchain), a consensus protocol is implemented in the public blockchain

network. Examples of consensus protocols include proof-of-work (POW) (e.g., implemented in the some crypto-currency networks), proof-of-stake (POS), and proof-of-authority (POA).

[0026] In general, a private blockchain network may be provided for a particular entity, which centrally controls read and write permissions. The entity controls which nodes are able to participate in the blockchain network. Consequently, private blockchain networks are generally referred to as permissioned networks that place restrictions on who is allowed to participate in the network, and on their level of participation (e.g., only in certain transactions). Various types of access control mechanisms can be used (e.g., existing participants vote on adding new entities, a regulatory authority can control admission).

[0027] In general, a consortium blockchain network may be private among the participating entities. In a consortium blockchain network, the consensus process is controlled by an authorized set of nodes, one or more nodes being operated by a respective entity (e.g., a financial institution, insurance company). For example, a consortium of ten (10) entities (e.g., financial institutions, insurance companies) can operate a consortium blockchain network, each of which operates at least one node in the consortium blockchain network. Accordingly, the consortium blockchain network can be considered a private network with respect to the participating entities. In some examples, each entity (node) must sign every block in order for the block to be valid, and added to the blockchain. In some examples, at least a sub-set of entities (nodes) (e.g., at least 7 entities) must sign every block in order for the block to be valid, and added to the blockchain.

[0028] FIG. 1 illustrates a schematic diagram of a blockchain system 100, according to an embodiment. Referring to FIG. 1, the blockchain system 100 may

include a plurality of nodes, e.g., nodes 102-110, configured to operate on a blockchain 120. The nodes 102-110 may form a network 112, such as a peer-to-peer (P2P) network. Each of the nodes 102-110 may be a computing device, such as a computer or a computer system, configured to store a copy of the blockchain 120, or may be software running on the computing device, such as a process or an application. Each of the nodes 102-110 may have a unique identifier.

[0029] The blockchain 120 may include a growing list of records in the form of data blocks, such as blocks B1-B5 in FIG. 1. Each of the blocks B1-B5 may include a timestamp, a cryptographic hash of a previous block, and data of the present block, which may be transactions such as monetary transactions. For example, as illustrated in FIG. 1, block B5 may include a timestamp, a cryptographic hash of block B4, and transaction data of block B5. Also, for example, a hashing operation may be performed on the previous block to generate the cryptographic hash of the previous block. The hashing operation may convert inputs of various lengths into cryptographic outputs of a fixed length through a hash algorithm, such as SHA-256.

[0030] The nodes 102-110 may be configured to perform an operation on the blockchain 120. For example, when a node, e.g., the node 102, wants to store new data onto the blockchain 120, that node may generate a new block to be added to the blockchain 120 and broadcast the new block to other nodes, e.g., the nodes 104-110, in the network 112. Based on legitimacy of the new block, e.g., validity of its signature and transactions, the other nodes may determine to accept the new block, such that the node 102 and the other nodes may add the new block to their respective copies of the blockchain 120. As this process repeats, more and more blocks of data may be added to the blockchain 120.

[0031] FIG. 2 illustrates a schematic diagram of a computing device 200 for

implementing a node, e.g., the node 102 (FIG. 1), in a blockchain system, according to an embodiment. Referring to FIG. 2, the computing device 200 may include a communication interface 202, a processor 204, and a memory 206.

[0032] The communication interface 202 may facilitate communications between the computing device 200 and devices implementing other nodes, e.g., nodes 104-110 (FIG. 1), in the network. In some embodiments, the communication interface 202 is configured to support one or more communication standards, such as an Internet standard or protocol, an Integrated Services Digital Network (ISDN) standard, etc. In some embodiments, the communication interface 202 may include one or more of a Local Area Network (LAN) card, a cable modem, a satellite modem, a data bus, a cable, a wireless communication channel, a radio-based communication channel, a cellular communication channel, an Internet Protocol (IP) based communication device, or other communication devices for wired and/or wireless communications. In some embodiments, the communication interface 202 may be based on public cloud infrastructure, private cloud infrastructure, hybrid public/private cloud infrastructure.

[0033] The processor 204 may include one or more dedicated processing units, application-specific integrated circuits (ASICs), field-programmable gate arrays (FPGAs), or various other types of processors or processing units. The processor 204 is coupled with the memory 206 and is configured to execute instructions stored in the memory 206.

[0034] The memory 206 may store processor-executable instructions and data, such as a copy of the blockchain 120 (FIG. 1). The memory 206 may include any type of volatile or non-volatile memory devices, or a combination thereof, such as a static random-access memory (SRAM), an electrically erasable programmable read-only memory (EEPROM), an erasable programmable read-only memory (EPROM), a

programmable read-only memory (PROM), a read-only memory (ROM), a magnetic memory, a flash memory, or a magnetic or optical disk. When the instructions in the memory 206 are executed by the processor 204, the computing device 200 may perform an operation on the blockchain 120.

[0035] Users of a blockchain system, e.g., the blockchain system 100, may utilize the blockchain system 100 to record various types of information. For example, in some embodiments, users may utilize the blockchain system 100 to record information regarding assets owned, managed, or traded by them. The assets may include, e.g., artworks, music, domain names, trading cards, collectibles, invoices, purchase orders, monetary assets, personal properties, real properties, intellectual properties, contracts, and the like.

[0036] In some embodiments, information regarding an asset may be represented by an identifier and a metadata field recorded on the blockchain system 100. The identifier and the metadata field may form a unit of data represented as, e.g., $\{ID: Metadata\}$, referred to as a token. In some embodiments, the $Metadata$ may include a set of attributes and their corresponding values. For example, if the asset represented is an artwork, the attributes may include, e.g., the name of the artist who created the artwork, the date of creation, a thumbprint of the artwork, an InterPlanetary File System (IPFS) link of the artwork, and the like. In another example, if the asset represented is an invoice or a purchase order, the attributes may include, e.g., the identity of the buyer, the identity of the seller, an amount of the invoice or the purchase, the goods or services purchased, and the like. In some embodiments, the asset represented may be unique and not interchangeable (i.e., a non-fungible asset). In such embodiments, the token representing the asset may be referred to as a non-fungible token (NFT).

[0037] There may be instances where a user, e.g., the owner of an NFT, may want to protect the plaintext values of the attributes or otherwise keep the plaintext values of the attributes hidden from other users of the blockchain system 100. In some embodiments, to protect the plaintext values of the attributes, the owner may encrypt the plaintext values of the attributes (e.g., using the owner's public key) so that only encrypted attribute values are recorded in the $Metadata$ field of the NFT. However, if the owner later sells the NFT to a buyer, the buyer may not be able to decrypt the encrypted attribute values without obtaining the secrete key from the owner. Also, for security reasons, the owner may not be willing to share its secrete key with the buyer. Furthermore, as the NFT continues to be bought and sold to subsequent buyers, it may become difficult to keep track of the keys required to decrypt the encrypted attribute values.

[0038] In some embodiments, instead of encrypting the attribute values, the owner may calculate a privacy-preserving representation of the attribute values and record the privacy-preserving representation of the attribute values as the $Metadata$ of the NFT. For example, if the privacy-preserving representation of the attribute values is denoted as PPR, the owner may record the NFT as $\{ID: PPR\}$.

[0039] In some embodiments, the owner may calculate the privacy-preserving representation of the attribute values by calculating a Merkle tree root hash according to a function. In some embodiments, the function may include a Merkle tree root hash function defined based on the attribute values. A Merkle tree refers to a tree where every leaf node is labelled with a cryptographic hash of a data block, and every non-leaf node is labelled with a cryptographic hash of the labels of its child nodes. For illustrative purposes, a Merkle tree is depicted in FIG. 3, where leaf nodes 302-308 are labelled with cryptographic hashes of data blocks L1-L4, respectively. Also, for

illustrative purposes, non-leaf nodes 310-314 are labelled with cryptographic hashes of the labels of their child nodes. Therefore, the non-leaf node 310 is labelled with the cryptographic hash of the labels of its child nodes, Hash 0-0 and Hash 0-1. Similarly, the non-leaf node 312 is labelled with the cryptographic hash of the labels of its child nodes, Hash 1-0 and Hash 1-1, and the non-leaf node 314 (which is also the Merkle tree root in this example) is labelled with the cryptographic hash of the labels of its child nodes, Hash 0 and Hash 1. In some embodiments, the owner may store the Merkle tree in a storage device. The storage device used for storing the Merkle tree may be implemented in various manners. In some embodiments, the owner may store the Merkle tree in a local data storage device. In some embodiments, the owner may store the Merkle tree on one or more cloud-based storage devices, distributed storage devices, or the like.

[0040] In some embodiments, to calculate the privacy-preserving representation of the attribute values, the owner may calculate the hashes of the attribute values and store the calculated hashes on the leaf nodes of a Merkle tree. For illustrative purposes, suppose the attribute values can be represented generally as:

$$\begin{cases} attribute_1 : value_1 \\ attribute_2 : value_2 \\ \quad\quad ... \\ attribute_N : value_N \end{cases},$$

where $\{attribute_i : value_i\}$ represents the value $value_i$ of a particular attribute $attribute_i$, then the Merkle tree with the calculated hashes stored therein may be depicted as shown in FIG. 4.

[0041] For example, as shown in FIG. 4, the values of the Merkle tree leaf nodes may be calculated based on the hashes of the different attribute-value pairs, $attribute_1 : value_1$ through $attribute_N : value_N$. The values of the non-leaf nodes may

be calculated as the hashes of their corresponding child nodes. Also shown in FIG. 4 is a cryptographic nonce (e.g., an arbitrary number) that may be used in some embodiments to randomize the root hash. Randomizing the root hash in this manner may help prevent other users from guessing the values of the attributes through trial and error.

[0042] For illustrative purposes, suppose the owner wants to record an NFT that represents a non-fungible asset on the blockchain system 100, and further suppose that the asset is associated with three attributes (e.g., the name of the artist who created the artwork, the date of creation, and a thumbprint of the artwork). FIG. 5 is a schematic diagram depicting a Merkle tree with hashes of the three attributes plus a nonce stored therein. As shown in FIG. 5, values of the leaf nodes, denoted as Hash 1, Hash 2, and Hash 3, are calculated based on the hashes of the different attribute-value pairs $attribute_1: value_1$ through $attribute_3: value_3$. The value of the remaining leaf node, denoted as Hash 4, is calculated based on the hash of the nonce. The values of the non-leaf nodes are calculated as hashes of their respective child nodes. Accordingly, the value of the root node, which may be referred to as the Merkle tree root hash, can be calculated. Similarly, FIG. 6 is a schematic diagram depicting a Merkle tree with hashes of six attributes plus a nonce stored therein, and the values of the non-leaf nodes depicted in FIG. 6, including the value of the root node, may be calculated accordingly, as described above.

[0043] In some embodiments, once the Merkle tree root hash is calculated, the owner may use the Merkle tree root hash as the privacy-preserving representation of the attribute values and record the privacy-preserving representation of the attribute values as the $Metadata$ of the NFT on the blockchain system 100. In this manner, the owner may keep the plaintext values of the attributes private, and the other users of

the blockchain system 100 may not be able to obtain the plaintext values of the attributes by reading the $Metadata$ of the NFT recorded on the blockchain system 100, thus preserving the privacy of the NFT.

[0044] Furthermore, recording the privacy-preserving representation of the attribute values as the $Metadata$ of the NFT may also support range verification of one or more attribute values associated with the asset represented by the NFT.

[0045] For example, in some embodiments, the owner may want to prove to a verifier (e.g., a buyer of the NFT or a third party) that the NFT recorded on the blockchain represents an asset having an attribute $attribute_i$ with a value $value_i$ that satisfies a condition without revealing the value $value_i$ of the attribute $attribute_i$ to the verifier. For example, the owner may want to prove to the verifier that the value $value_i$ of the $attribute_i$ is greater than l (e.g., $l < value_i$), greater than or equal to l (e.g., $l \leq value_i$), less than u (e.g., $value_i < u$), less than or equal to u (e.g., $value_i \leq u$), etc. In some instances, the owner may want to prove to the verifier that $l < value_i < u$, $l \leq value_i \leq u$, or the like.

[0046] In some embodiments, to facilitate the range verification of the attribute value, the owner may record the Merkle tree as a binary Merkle tree such that a hash stored in a leaf node of the Merkle tree can be identified based on a publicly known root hash and an authentication path of that leaf node. In some embodiments, the authentication path of a particular leaf node may include siblings of all nodes on the path from that particular leaf node to the Merkle tree root. FIG. 7 is a schematic diagram depicting an authentication path of a leaf node 702, according to an embodiment. As shown in FIG. 7, the authentication path of the leaf node 702 includes a node 704 because the node 704 is a sibling of the leaf node 702. The authentication path also includes a node 706 because the node 706 is a sibling of node 708, which

is a node on the path from the leaf node 702 to a Merkle tree root 718. Similarly, the authentication path includes a node 710 because the node 710 is a sibling of a node 712, which is a node on the path from the leaf node 702 to the Merkle tree root 718. Likewise, the authentication path also includes a node 714 because the node 714 is a sibling of a node 716, which is a node on the path from the leaf node 702 to the Merkle tree root 718.

[0047] In some embodiments, the owner may obtain the hashes $p_1, p_2, \dots, p_N$ stored in the nodes 704, 706, …, 714 on the authentication path of the leaf node 702 and use the obtained hashes $p_1, p_2, \dots, p_N$ to generate a proof for proving that the value of $attribute_i$ satisfies a condition (e.g., $l < value_i < u$) without revealing the value of $attribute_i$ to the verifier.

[0048] In some embodiments, the owner may provide a zero-knowledge proof to the verifier for proving that the value of $attribute_i$ satisfies the condition (e.g., $l < value_i < u$). Zero-knowledge proofs refer to techniques that allow a prover to prove to a verifier that a statement is true without revealing any information beyond the validity of the statement itself. For example, the owner may prove to the verifier that the owner knows a secret input w such that for a public input x, a certain relationship between x and w holds true.

[0049] In some embodiments, the owner and the verifier may agree to implement zero-knowledge proof techniques such as Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK) or the like, e.g., as disclosed in Gennaro R., Gentry C., Parno B., Raykova M. (2013) Quadratic Span Programs and Succinct NIZKs without PCPs. In: Johansson T., Nguyen P.Q. (eds) Advances in Cryptology – EUROCRYPT 2013. EUROCRYPT 2013. Lecture Notes in Computer Science, vol 7881. Springer, which is herein incorporated by reference in its entirety.

In some embodiments, the owner and the verifier may agree to implement zk-SNARK that includes a key generator algorithm G. The key generator algorithm G may take a secret parameter λ and a program C as input and generate two publicly available keys, a proving key pk and a verification key vk, as output. In some embodiments, the proving key pk and the verification key vk may be generated as public parameters that only need to be generated once in a setup phase for a given program $C(x, w)$, where $C(x, w)$ returns true if the owner provides the correct values of x and w. One of ordinary skill in the art will understand that the setup phase may be executed by a trusted party, or by multiple independent parties collaboratively using a multi-party computation.

[0050] In some embodiments, zk-SNARK may also include a prover algorithm P and a verifier algorithm V. In some embodiments, the prover algorithm P may take the proving key pk, the public input x, and the secret input w as input and generate a zero-knowledge proof $\pi = P(pk, x, w)$ that can prove to the verifier that the owner knows the secret input w and that the secret input w makes $C(x, w) == true$. The verifier algorithm V may take the verification key vk, the public input x, and the proof π as input and compute $V(vk, x, \pi)$. In some embodiments, $V(vk, x, \pi)$ may be defined to return true if the proof π is acceptable, which means that the owner knows the secret input w and that the secret input w makes $C(x, w) == true$. Otherwise, if the proof π is unacceptable, the verifier algorithm $V(vk, x, \pi)$ may return false.

[0051] Continuing with the example above, in some embodiments, the owner may generate a zero-knowledge proof π, in which the owner may set the public input x to include the attribute $attribute_i$, the value of l, the value of u, and the $Metadata$ of the NFT, i.e., $x = (attribute_i, l, u, Metadata)$, and set the secret input w to include the value $value_i$ of the $attribute_i$ and the hashes $p_1, p_2, \dots, p_N$ obtained from the

authentication path, i.e., $w = (value_i, p_1, p_2, \dots, p_N)$. It is noted that the plaintext values of the attribute $attribute_i$ and the other attributes associated with the asset are not disclosed to the verifier to preserve privacy.

[0052] In some embodiments, the owner may set the value of x by concatenating $(attribute_i, l, u, Metadata)$ together and set the value of w by concatenating $(value_i, p_1, p_2, \dots, p_N)$ together. The owner may also define $C(x, w)$ such that $C(x, w)$ returns true if and only if:

$$Metadata = \mathrm{H}\langle \dots \mathrm{H}\{\mathrm{H}[\mathrm{H}(p_1 || \mathrm{H}(attribute_i || value_i)) || p_2] || p_3\} \dots || p_N \rangle$$

and

$$l < value_i < u$$

where $H(\,)$, $H[\,]$, $H\{\,\}$, and $H\langle\ \ \rangle$ represent the same hash function utilized to calculate the hashes recorded on the Merkle tree, as described above. In this manner, the owner may generate the proof π using the secret and public inputs as well as the proving key to prove to the verifier that the owner is in possession of the secret input w that makes the above condition true. For example, if the relationship between the public input x and the secret input w can be defined so that $C(x, w)$ holds true, then a proving key pk and a verification key vk may be setup, and the owner may generate the proof π using the prover algorithm P to prove to the verifier that $Metadata = \mathrm{H}\langle \dots \mathrm{H}\{\mathrm{H}[\mathrm{H}(p_1 || \mathrm{H}(attribute_i || value_i)) || p_2] || p_3\} \dots || p_N \rangle$ and $l < value_i < u$.

[0053] In some embodiments, the verifier may verify the proof π using the public input x and the verification key. Continuing with the example above, the verifier may verify the proof using the verifier algorithm V to determine whether $V(vk, x, \pi)$ returns true, which means the owner knows the secret input w and that the secret input w makes $C(x, w) == true$. Otherwise, if the proof π is unacceptable, the verifier algorithm $V(vk, x, \pi)$ may return false. In this manner, if $C(x, w) == true$, the verifier

may accept the condition $Metadata = \mathrm{H}\langle \ldots \mathrm{H}\{\mathrm{H}[\mathrm{H}(p_1||\mathrm{H}(attribute_i||value_i))||p_2]||p_3\} \ldots ||p_N\rangle$ and $l < value_i < u$ as true, which may indicate that the value of $attribute_i$ satisfies the condition that $l < value_i < u$. On the other hand, if the equation does not hold true, the verifier may refuse to accept the proof. In this manner, if the owner indicates to the verifier that the asset represented by the NFT is an asset that has an attribute $attribute_i$ with a value $value_i$ that satisfies a condition that $l < value_i < u$, and if the owner can prove to the verifier that the value of $attribute_i$ indeed satisfies the condition that $l < value_i < u$, the verifier may accept the NFT as representing the asset as indicated by the owner. Otherwise, the verifier may refuse to accept the NFT and refuse to carry out any transactions associated with that NFT.

[0054] It is to be understood that the equation, the proving key, and the verification key described above are merely presented as examples and are not meant to be limiting. It is contemplated that the relationship between the public input x and the secret input w can be defined using various types of equations, and that the proving key and the verification key may be defined accordingly based on the relationship between the public input x and the secret input w. In some embodiments, the proof may also be translated into sigma protocols, which can be transformed into non-interactive zero-knowledge proofs of knowledge using Fiat-Shamir transform.

[0055] It is also to be understood that the condition $l < value_i < u$ referenced above is merely presented as an example and is not meant to be limiting. It is contemplated that various types of condition, including, e.g., $l < value_i$, $value_i < u$, $l \leq value_i \leq u$, and the like, can be defined and proved using the verification process described above. It is also contemplated that the verification process described above can support verifications of more than one conditions. For example, if the condition

$l < value_i < u$ can be expressed generally as $Condition_1(value_i)$, the owner may use the verification process described above to prove additional conditions $Condition_2(value_i)$, $Condition_3(value_i)$, . . . by defining the arithmetic circuit $a(x)w(x) = b(x)c(x)$ such that the equation holds true if and only if:

$$Metadata = \mathrm{H}\langle \dots \mathrm{H}\{\mathrm{H}[\mathrm{H}(p_1||\mathrm{H}(attribute_i||value_i))||p_2]||p_3\} \dots ||p_N\rangle$$

and

$$Condition_1(value_i)$$

and

$$Condition_2(value_i)$$

and

$$Condition_3(value_i)$$

…

[0056] The Owner may set the public input x to include the values of publicly accessible variables referenced in the conditions (e.g., the values of l and u in $Condition_1$) and set the secret input w to include the value $value_i$ of the $attribute_i$ and the hashes $p_1, p_2, \dots, p_N$, as described above. In this manner, the owner may generate the proof π by calculating the polynomial $a(x)w(x)$ to prove to the verifier that $Metadata = \mathrm{H}\langle \dots \mathrm{H}\{\mathrm{H}[\mathrm{H}(p_1||\mathrm{H}(attribute_i||value_i))||p_2]||p_3\} \dots ||p_N\rangle$ and that $Condition_1(value_i)$, $Condition_2(value_i)$, $Condition_3(value_i)$, . . . are all satisfied.

[0057] In some embodiments, the verifier may be a user of the blockchain system 100. In some embodiments, the owner may provide the proof π to the verifier through an off-chain communication channel, which may include various types of wired or wireless communication channels, such as the Internet, electronic mails, file transfer applications, text messages, and the like. Upon receiving the proof π from the owner,

the verifier may invoke a verification program that takes the proof π as input and performs the verification process described above. In some embodiments, the verification program invoked by the verifier may be carried out on a computing device that does not participate in the blockchain system 100. In some embodiments, the verifier may invoke one or more smart contracts executing on the blockchain system 100 to perform the verification.

[0058] Smart contracts are computer protocols implemented in the form of computer code that are incorporated into the blockchain, to facilitate, verify, or enforce the negotiation or performance of agreed terms or conditions. For example, users of the blockchain system 100 may program agreed terms into a smart contract using a programming language, such as C++, Java, Solidity, Python, etc., and when the terms are met, the smart contract may be automatically executed on the blockchain system 100. Also for example, the smart contract may include a plurality of subroutines or functions, each of which may be a sequence of program instructions that performs a specific task. The smart contract may be operational code that is fully or partially executed without human interaction. In some embodiments, the blockchain system 100, through one or more smart contracts executing on the blockchain system 100, may receive as input the proof π generated by the owner. In this manner, the blockchain system 100, through one or more smart contracts executing thereon, may perform the verification process described above.

[0059] In some embodiments, the owner may provide the proof π directly to a smart contract executing on the blockchain system 100 and utilize the smart contract as the verifier to verify the attribute value as described above. For example, the owner may submit a transaction to the blockchain system 100 and include the proof π as a part of the transaction. Upon receiving the transaction, the blockchain system 100,

through one or more smart contracts executing thereon, may perform the verification process described above. If the condition(s) can be verified, the blockchain system 100 may record the transaction on the blockchain system 100, allowing other users to see that the condition(s) have been verified already. On the other hand, if the condition(s) cannot be verified, the blockchain system 100 may refuse to record the transaction on the blockchain system 100.

[0060] FIG. 8 illustrates a flow chart of a method 800 for verifying an attribute value of an asset, according to an embodiment. For illustrative purposes, a Blockchain, e.g., the blockchain 120 (FIG. 1), is depicted in FIG. 8. The Blockchain may be implemented to support various types of users, or parties, including, e.g., individuals, businesses, banks, financial institutions, as well as other types of companies, organizations, and the like. For illustrative purposes, a user, referred to as an Owner, is depicted in FIG. 8, and the operations of the Owner described below are performed by a computing device of the Owner.

[0061] At step 802, the Owner may create (or "mint") a token representing an asset owned by the Owner. As described above, in some embodiments, the asset may be represented by a token that includes an identifier and a metadata field, $\{ID: Metadata\}$. In some embodiments, the asset represented may be unique and not interchangeable (i.e., a non-fungible asset). In such embodiments, the token representing the asset may be referred to as a non-fungible token (NFT).

[0062] At step 804, the Owner may calculate the $Metadata$ of the token. As described above, the Owner may calculate a privacy-preserving representation of a set of attributes and record the privacy-preserving representation as the $Metadata$ of the token. In some embodiments, the Owner may calculate the privacy-preserving representation by calculating a Merkle tree root hash according to a function, and the

function may be defined based on a value associated with each attribute of the set of attributes. In some embodiments, the function may include a Merkle tree root hash function that calculates a Merkle tree root hash based on the value associated with each attribute of the set of attributes, and in some embodiments, the value associated with each attribute of the set of attributes may have a corresponding hash stored on a leaf node of the Merkle tree. Furthermore, in some embodiments, the function may also be defined based on a cryptographic nonce. In some embodiments, the cryptographic nonce may have a corresponding hash stored on a node of the Merkle tree.

[0063] At step 806, the Owner may record the token on the Blockchain. As described above, in some embodiments, the Owner may record the privacy-preserving representation of the set of attributes on the Blockchain as the $Metadata$ of the token. In some embodiments, the Owner may encrypt the ID of the recorded token. In some embodiments, the encryption of the ID may be optional and is not required.

[0064] At step 808, the Owner may generate a proof for proving that a value $value_i$ associated with a particular attribute $attribute_i$ of the set of attributes satisfies a condition. In some embodiments, the condition may include a comparison of the value $value_i$ against at least one specified value. For example, the condition may include a comparison indicating $l < value_i$, $l \leq value_i$, $value_i < u$, $value_i \leq u$, $l < value_i < u$, $l \leq value_i \leq u$, or the like. As described above, in some embodiments, the Owner may generate the proof as a zero-knowledge proof.

[0065] At step 810, the Owner may provide the proof to a Verifier to facilitate a verification process. As described above, in some embodiments, the Verifier may be a user of the Blockchain. In some embodiments, the Verifier may be a smart contract deployed in the Blockchain that can be utilized to perform the verification process. In

some embodiments, the Verifier may be a user of the Blockchain, and the user may choose to invoke a smart contract deployed in the Blockchain to perform the verification process. While specific implementations may vary, the Verifier may, at step 812, obtain the token recorded on the Blockchain and carry out the verification process described above at step 814. If the Verifier determines that the proof provided by the Owner is acceptable, the Verifier may accept the proof as indicating that the token recorded on the Blockchain represents an asset that has an attribute $attribute_i$ with a value $value_i$ that satisfies the condition. On the other hand, if the Verifier determines that the proof provided by the Owner is unacceptable, the Verifier may refuse to accept the proof as indicating that the token recorded on the Blockchain represents an asset that has an attribute $attribute_i$ with a value $value_i$ that satisfies the condition and refuse to carry out any transactions associated with the token. In some embodiments, the verification process may be carried out without requiring the Owner to reveal the values of the attributes associated with the asset represented by the token, including $value_i$ of $attribute_i$ referenced in the example above.

[0066] FIG. 9 illustrates a flow chart of a method 900 for verifying an attribute value of an asset, according to an embodiment. The method 900 may be performed by a computing device communicating with a blockchain system, e.g., the blockchain system 100 (FIG. 1). The blockchain system may include a plurality of nodes, e.g., nodes 102-110, configured to operate on a blockchain 120. The blockchain 120 may be implemented as the Blockchain in the examples described above. The computing device may or may not be a node of the blockchain system.

[0067] A user may generate a privacy-preserving token based on steps 902 and 904. At step 902, the user may be the creator or the author of an asset, and the user may utilize a computing device (e.g., the computing device 200) to calculate a privacy-

preserving representation of a set of attributes associated with an asset by calculating a Merkle tree root hash based on a value associated with each attribute of the set of attributes. In some embodiments, the user may calculate the privacy-preserving representation according to a function defined based on the value associated with each attribute of the set of attributes. In some embodiments, the function may include a Merkle tree root hash function that calculates a Merkle tree root hash based on the value associated with each attribute of the set of attributes, and in some embodiments, the value associated with each attribute of the set of attributes may have a corresponding hash stored on a leaf node of the Merkle tree. Furthermore, in some embodiments, the function may be further defined based on a cryptographic nonce. In some embodiments, the cryptographic nonce may have a corresponding hash stored on a node of the Merkle tree.

[0068] At step 904, the user may record the privacy-preserving representation of the set of attributes on the blockchain system. In some embodiments, the user may utilize the computing device to submit the privacy-preserving representation of the set of attributes to the blockchain system for recordation. Upon receiving the privacy-preserving representation submitted by the user, the blockchain system may record the privacy-preserving representation on a blockchain of the blockchain system. In some embodiments, the blockchain system may record the privacy-preserving representation as metadata of a token representing the asset. In some embodiments, the blockchain system may also record an identifier of the token, and in some embodiments, the identifier of the token may be encrypted.

[0069] At step 906, the user may utilize the computing device to generate a proof for proving that the value associated with a particular attribute of the set of attributes satisfies a condition. Continuing with the examples above, if the particular

attribute is $attribute_i$, the user may identify a particular leaf node (e.g., the leaf node 702, FIG. 7) that stores the hash corresponding to the value associated with the particular attribute $attribute_i$, and identify an authentication path of the particular leaf node as described above. The user may also obtain the hashes (e.g., $p_1, p_2, \dots, p_N$ in FIG. 7) stored in the nodes on the authentication path and generate a proof based on the obtained hashes. In some embodiments, the authentication path of the particular leaf node may include siblings of all nodes on a path from the particular leaf node to the Merkle tree root, and in some embodiments, the user may form a secrete input based on the value associated with the particular attribute and the obtained hashes, and generate the proof as a zero-knowledge proof based on the secrete input, as described above.

[0070] At step 908, the user may provide the proof to a verifier to facilitate a verification of whether the value associated with the particular attribute of the set of attributes satisfies the condition. In some embodiments, the condition may include a comparison of the value $value_i$ against at least one specified value. For example, the condition may include a comparison indicating $l < value_i$, $l \leq value_i$, $value_i < u$, $value_i \leq u$, $l < value_i < u$, $l \leq value_i \leq u$, or the like. In some embodiments, the verification may be performed without revealing the values of the set of attributes, including the value $value_i$ of the particular $attribute_i$ referenced above. In some embodiments, the user who performs the steps 906 and 908 may be the same as the user who performed the steps 902 and 904. For example, in some embodiments, the user may be the creator of the asset, and the user may perform the steps 906 and 908 after recording the metadata of the token on the blockchain system. Alternatively, in some embodiments, the user who performs the steps 906 and 908 may be different from the user who performed the steps 902 and 904. For example, the creator of the

asset may have transferred the ownership of the token to a different owner. Therefore, in some embodiments, the steps 902 and 904 may be performed by the creator of the asset, and the steps 906 and 908 may be performed by the owner of the asset, who may or may not be the same as the creator.

[0071] In some embodiments, the verifier may be a user of the blockchain system. In some embodiments, the verifier may be a smart contract deployed in the blockchain system. At step 910, the verifier may receive a proof for proving that the value associated with a particular attribute of the set of attributes satisfies a condition, and at step 912, the verifier may verify whether the value associated with the particular attribute of the set of attributes satisfies the condition without knowing the value associated with at least one other attribute of the set of attributes, as described above. In some embodiments, if the verifier determines that the proof provided by the user is acceptable, the verifier may accept the proof as indicating that the value associated with the particular attribute of the set of attributes satisfies the condition. On the other hand, if the verifier determines that the proof provided by the user is unacceptable, the verifier may refuse to accept the token and refuse to carry out any transactions associated with that token.

[0072] FIG. 10 is a block diagram of an apparatus 1000 for verifying an attribute value of an asset, according to an embodiment. The apparatus 1000 may be an implementation of a software process, and may correspond to the method 900 (FIG. 9). Referring to FIG. 10, the apparatus 1000 may include a calculation module 1002, a recording module 1004, a generation module 1006, and an interface module 1008.

[0073] The calculation module 1002 may calculate a privacy-preserving representation of a set of attributes associated with an asset, as described above, and

provide the calculated privacy-preserving representation to the recording module 1004. The recording module 1004 may communicate with a data storage system, e.g., a blockchain system, to record the privacy-preserving representation of the set of attributes on the date storage system. The generation module 1006 may generate a proof for proving that the value associated with a particular attribute of the set of attributes satisfies a condition. The generation module 1006 may provide the proof to the interface module 1008, which may provide the proof to a verifier through an interface (e.g., a user interface or an application programming interface) to facilitate a verification of whether the value associated with the particular attribute of the set of attributes satisfies the condition. In some embodiments, the verification may be performed without revealing the values of the attributes associated with the asset, including the value of the particular attribute referenced above.

[0074] Each of the above-described modules may be implemented as software, or hardware, or a combination of software and hardware. For example, each of the above described modules may be implemented using a processor executing instructions stored in a memory. Also, for example, each the above described modules may be implemented with one or more application specific integrated circuits (ASICs), digital signal processors (DSPs), digital signal processing devices (DSPDs), programmable logic devices (PLDs), field programmable gate arrays (FPGAs), controllers, micro-controllers, microprocessors, or other electronic components, for performing the described methods. Further for example, each of the above described modules may be implemented by using a computer chip or an entity, or implemented by using a product having a certain function. In one embodiment, the apparatus 1000 may be a computer, and the computer may be a personal computer, a laptop computer, a cellular phone, a camera phone, a smartphone, a personal digital

assistant, a media player, a navigation device, an email receiving and sending device, a game console, a tablet computer, a wearable device, or any combination of these devices.

[0075] For an implementation process of functions and roles of each module in the apparatus 1000, references can be made to corresponding steps in the above-described methods. Details are omitted here for simplicity.

[0076] In some embodiments, a computer program product may include a non-transitory computer-readable storage medium having computer-readable program instructions thereon for causing a processor to carry out the above-described methods.

[0077] The computer-readable storage medium may be a tangible device that can store instructions for use by an instruction execution device. The computer-readable storage medium may be, for example, but is not limited to, an electronic storage device, a magnetic storage device, an optical storage device, an electromagnetic storage device, a semiconductor storage device, or any suitable combination of the foregoing. A non-exhaustive list of more specific examples of the computer-readable storage medium includes the following: a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM), a static random access memory (SRAM), a portable compact disc read-only memory (CD-ROM), a digital versatile disk (DVD), a memory stick, a floppy disk, a mechanically encoded device such as punch-cards or raised structures in a groove having instructions recorded thereon, and any suitable combination of the foregoing.

[0078] The computer-readable program instructions for carrying out the above-described methods may be assembler instructions, instruction-set-architecture (ISA)

instructions, machine instructions, machine dependent instructions, microcode, firmware instructions, state-setting data, or source code or object code written in any combination of one or more programming languages, including an object oriented programming language, and conventional procedural programming languages. The computer-readable program instructions may execute entirely on a computing device as a stand-alone software package, or partly on a first computing device and partly on a second computing device remote from the first computing device. In the latter scenario, the second, remote computing device may be connected to the first computing device through any type of network, including a local area network (LAN) or a wide area network (WAN).

[0079] The computer-readable program instructions may be provided to a processor of a general-purpose or special-purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the above-described methods.

[0080] The flow charts and diagrams in the figures illustrate the architecture, functionality, and operation of possible implementations of devices, methods, and computer program products according to various embodiments of the specification. In this regard, a block in the flow charts or diagrams may represent a software program, segment, or portion of code, which comprises one or more executable instructions for implementing specific functions. It should also be noted that, in some alternative implementations, the functions noted in the blocks may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block

of the diagrams and/or flow charts, and combinations of blocks in the diagrams and flow charts, may be implemented by special purpose hardware-based systems that perform the specified functions or acts, or combinations of special purpose hardware and computer instructions.

[0081] It is appreciated that certain features of the specification, which are, for clarity, described in the context of separate embodiments, may also be provided in combination in a single embodiment. Conversely, various features of the specification, which are, for brevity, described in the context of a single embodiment, may also be provided separately or in any suitable subcombination or as suitable in any other described embodiment of the specification. Certain features described in the context of various embodiments are not essential features of those embodiments, unless noted as such.

[0082] Although the specification has been described in conjunction with specific embodiments, many alternatives, modifications and variations will be apparent to those skilled in the art. Accordingly, the following claims embrace all such alternatives, modifications and variations that fall within the terms of the claims.

What Is Claimed Is:

1. A computer-implemented method for verifying an attribute value of an asset represented by a privacy-preserving token, wherein a privacy-preserving representation of a set of attributes associated with the asset is calculated by calculating a Merkle tree root hash based on a value associated with each attribute of the set of attributes and is recorded on a blockchain system as metadata of the token, and the value associated with each attribute of the set of attributes has a corresponding hash stored on a leaf node of the Merkle tree, the method comprising:

generating a proof for proving that the value associated with a particular attribute of the set of attributes satisfies a condition; and

providing the proof to a verifier to facilitate a verification of whether the value associated with the particular attribute of the set of attributes satisfies the condition without revealing the value associated with the particular attribute;

wherein generating the proof comprises:

forming a public input based on the particular attribute, at least one specified value, and the metadata of the token;

forming a secret input based on the value associated with the particular attribute and one or more hashes obtained from one or more nodes on an authentication path of a particular leaf node that stores the hash corresponding to the value associated with the attribute; and

generating the proof as a zero-knowledge proof based on the public input and the secret input.

2. A computer-implemented method for verifying an attribute value of an asset represented by a privacy-preserving token, wherein a privacy-preserving representation of a set of attributes associated with the asset is calculated by calculating a Merkle tree root hash based on a value associated with each attribute of the set of attributes and is recorded on a blockchain system as metadata of the token, and the value associated with each attribute of the set of attributes has a corresponding hash stored on a leaf node of the Merkle tree, the method comprising:

receiving a proof for proving that the value associated with a particular attribute of the set of attributes satisfies a condition; and

verifying whether the value associated with the particular attribute of the set of attributes satisfies the condition without knowing the value associated with the particular attribute

wherein the proof is generated by:

forming a public input based on the particular attribute, at least one specified value, and the metadata of the token;

forming a secret input based on the value associated with the particular attribute and one or more hashes obtained from one or more nodes on an authentication path of a particular leaf node that stores the hash corresponding to the value associated with the attribute; and

generating the proof as a zero-knowledge proof based on the public input and the secret input.

3. The method of claim 1 or 2, wherein the condition comprises a comparison of the value associated with the particular attribute and the at least one specified value.

4. The method of any preceding claim, wherein the token comprises: a token identifier and the metadata of the token.

5. The method of any preceding claim, wherein the token is a non-fungible token.

6. The method of any preceding claim, wherein the privacy-preserving representation is calculated by calculating the Merkle tree root hash based on a cryptographic nonce.

7. The method of claim 6, wherein a hash corresponding to the cryptographic nonce is stored on a node of the Merkle tree.

8. The method of claim 1, further comprising:

identifying the particular leaf node that stores the hash corresponding to the value associated with the particular attribute of the set of attributes;

identifying the authentication path of the particular leaf node;

obtaining the one or more hashes stored in one or more nodes on the authentication path; and

generating the proof based on the one or more obtained hashes.

9. The method of any preceding claim, wherein the authentication path of the particular leaf node comprises siblings of all nodes on a path from the particular leaf node to a Merkle tree root.

10. A device for verifying an attribute value of an asset, comprising:

one or more processors; and

one or more computer-readable memories coupled to the one or more processors and having instructions stored thereon that are executable by the one or more processors to perform the method of any of claims 1 to 9.

11. An apparatus for verifying an attribute value of an asset, the apparatus comprising a plurality of modules for performing the method of any of claims 1 to 9.

12. A non-transitory computer-readable medium having stored therein instructions that, when executed by a processor of a device, cause the device to perform the method of any of claims 1 to 9.