

METHODS AND DEVICES FOR NON-FUNGIBLE TOKEN REMIXING BASED ON BLOCKCHAIN

TECHNICAL FIELD

[0001] The specification relates generally to computer technologies, and more particularly, to methods and devices for non-fungible token remixing based 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 or 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, or 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, or 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.

SUMMARY

[0007] In one aspect, a computer-implemented method for remixing non-fungible tokens (NFTs) includes: obtaining a first NFT of a first type and a second NFT of a second type based on a request of a user, the first NFT being stored on a blockchain and associated with a first identifier and first metadata, and the second NFT being stored on the blockchain and associated with a second identifier and second metadata; determining whether the user has remixing authorization of at least one of the first NFT or the second NFT; if the user has the remixing authorization of at least one of the first NFT or the second NFT, determining whether remixing is permissible for the first NFT and the second NFT based on the first NFT and the second NFT; if the remixing is permissible for the first NFT and the second NFT, generating an output identifier based on the first identifier and the second identifier, generating an output type based on the first type and the second type, and generating output metadata based on the first metadata and the second metadata; and generating an output NFT based on the output identifier, the output type, and the output metadata, and recording the output NFT on the blockchain.

[0008] In another aspect, a device for remixing NFTs includes: 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: obtain a first NFT of a first type and a second NFT of a second type based on a request of a user, the first NFT being stored on a blockchain and associated with a first identifier and first metadata, and the second NFT being stored on the blockchain and associated with a second identifier and second metadata; determine whether the user has remixing authorization of at least one of the first NFT or the second NFT; if the user has the remixing authorization of at least

one of the first NFT or the second NFT, determine whether remixing is permissible for the first NFT and the second NFT based on the first NFT and the second NFT; if the remixing is permissible for the first NFT and the second NFT, generate an output identifier based on the first identifier and the second identifier, generate an output type based on the first type and the second type, and generate output metadata based on the first metadata and the second metadata; and generate an output NFT based on the output identifier, the output type, and the output metadata, and recording the output NFT on the blockchain.

[0009] In still another aspect, a non-transitory computer-readable medium has stored therein instructions that, when executed by a processor of a device, cause the device to perform a method for remixing NFTs. The method includes: obtaining a first NFT of a first type and a second NFT of a second type based on a request of a user, the first NFT being stored on a blockchain and associated with a first identifier and first metadata, and the second NFT being stored on the blockchain and associated with a second identifier and second metadata; determining whether the user has remixing authorization of at least one of the first NFT or the second NFT; if the user has the remixing authorization of at least one of the first NFT or the second NFT, determining whether remixing is permissible for the first NFT and the second NFT based on the first NFT and the second NFT; if the remixing is permissible for the first NFT and the second NFT, generating an output identifier based on the first identifier and the second identifier, generating an output type based on the first type and the second type, and generating output metadata based on the first metadata and the second metadata; and generating an output NFT based on the output identifier, the output type, and the output metadata, and recording the output NFT on the blockchain.

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 flow chart of a method for remixing non-fungible tokens, according to an embodiment.

[0014] FIG. 4 is a block diagram of an apparatus for remixing non-fungible tokens, according to an embodiment.

DETAILED DESCRIPTION

[0015] Remixing of non-fungible tokens (NFTs), as used herein, may refer to creating or generating an output NFT (i.e., new NFT) based on two or more input NFTs (i.e., existing NFTs). For example, in a blockchain system, a first NFT may represent a first song created by a first creator, and a second NFT may represent a second song created by a second creator. Through NFT remixing, a third creator may produce a third song (represented by a third NFT) by sampling and remixing the first song and the second song under permissions granted by the first creator and the second creator. With such granted permissions, when the third NFT is generated, the first creator and the second creator may automatically receive royalty fees deducted from an account of the third creator, the royalty fees being defined by

a smart contract that is stored in the blockchain system. As another example, in a blockchain system, a first NFT may represent a first digital pet (e.g., a virtual dog) owned by a first player of a game, and a second NFT may represent a second digital pet owned by a second player of the game. Through NFT remixing, the first digital pet and the second digital pet may reproduce a digital offspring (e.g., a virtual puppy) in the game under permissions granted by the first player and the second player and under a condition that the game rules allow such reproduction (e.g., both the first and second digital pets are virtual dogs).

[0016] Embodiments of the specification provide methods and devices for remixing NFTs. The methods and devices may obtain a first NFT of a first type and a second NFT of a second type. The methods and devices may determine whether the user has remixing authorization of at least one of the first NFT or the second NFT. If so, the methods and devices may also determine whether remixing is permissible for the first NFT and the second NFT based on the first NFT and the second NFT. If so, the methods and devices may further generate an output NFT based on the first NFT and the second NFT by a remixing technique. In such a remixing technique, NFT identifiers, NFT types, and NFT metadata of the inputted first and second NFTs may be processed and outputted to form the output NFT.

[0017] Embodiments in the specification have one or more technical effects. In some embodiments, the methods and devices may determine whether the user has remixing authorization of at least one of the first NFT or the second NFT based on first metadata included in the first NFT or second metadata included in the second metadata. This allows the methods and devices to ensure security of input NFTs and prevent unauthenticated remixing. In some embodiments, the methods and devices may further determine whether the remixing is permissible for the first

NFT and the second NFT based on indicators included in the first NFT and the second NFT. This allows the methods and devices to further ensure security of the remixing and prevent unauthorized remixing. By doing so, payment of required fees (e.g., license fees) may be enforced during the remixing of the NFTs automatically. Furthermore, the provided technical solutions may allow users of a blockchain that supports NFT remixing to define their own remixing algorithms, thus making the provided technical solution robust, versatile, and user-friendly.

[0018] 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.

[0019] 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.

[0020] 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).

[0021] 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).

[0022] 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.

[0023] 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.

[0024] 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.

[0025] 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.

[0026] 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.

[0027] 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.

[0028] 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.

[0029] 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.

[0030] 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, or the like.

[0031] 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 artwork, the description of the artwork, the name of the artist who created the artwork, the signature of the artist, the date or timestamp of creation, a thumbprint of the artwork, an InterPlanetary File System (IPFS) link of the artwork, or 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, or 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).

[0032] In some embodiments, information regarding an asset may be represented by an identifier, a type (e.g., an enumeration-type data field), and a metadata field recorded on the blockchain system 100. The identifier, the type, and the metadata field may form a token represented as, e.g., $\{ID: TokenType: Metadata\}$. The *TokenType* may represent a type of the NFT, e.g., a song, a video, a paint, a photo, an invoice, a game character, a game prop, a purchase order, or the like. Depending on the *TokenType*, the *Metadata* may include different attributes or properties of the NFT arranged in various formats. In some embodiments, the type of the token may be recorded as a part of the identifier of the token (e.g., $ID = E000234$, indicating that the token is of token type “E”). In

another example, the type of the token may be recorded as an attribute in the metadata of the token. It is to be understood that the type of the token may be associated with the token in other manners as well.

[0033] By way of example, the *Metadata* may be a JSON object defined by a JSON schema. Table 1 presents an example JSON object for *Metadata* of an artwork-type NFT, which includes multiple attributes (e.g., a name of the artwork, a description of the artwork, a link to an original image of the artwork, an identity of the artist of the artwork, a signature of the artist, and a production date of the artwork) of the artwork-type NFT.

Line No.	
1	{
2	"title": "Metadata of Artwork NFT",
3	"type": "object",
4	"properties": {
5	"name": {
6	"type": "string",
7	"description": "artwork name"
8	},
9	"description": {
10	"type": "string",
11	"description": "artwork description"
12	},
13	"image": {
14	"type": "string",
15	"description": "a link to the original image"
16	},
17	"artist": {
18	"type": "string",
19	"description": "artist's identity/public key"
20	},
21	"signature": {
22	"type": "string",
23	"description": "artist's signature"
24	},
25	"timestamp": {

26	"type": "uint64",
27	"description": "the production date of the artwork"
28	}
29	}
30	}

Table 1 Example Metadata of Artwork-Type NFT

[0034] In some embodiments, a user, e.g., the current owner of first and second input NFTs (e.g., two original songs), may generate an output NFT based on the first and second input NFTs (e.g., to generate a remixed song based on two original songs). To provide a creator (e.g., a songwriter) of an input NFT with intellectual property rights, a mechanism of permission-based NFT remixing may be used. A creator, as used herein, may refer to an individual, a group of individuals, or a non-individual entity (e.g., a company) that creates an NFT for the first time. For example, if the creator permits NFT remixing, such a mechanism may permit loyalty fee automatically deducted from an account of the owner of the NFT who conducts the NFT remixing and paid to the creator. In some embodiments, such a permission-based NFT remixing mechanism may also allow NFT remixing for other users (e.g., a licensee) of the NFT. By way of example, a licensee of an input NFT may be previously authorized by the owner of the input NFT to conduct NFT remixing, and the NFT may be permitted by the creator for NFT remixing.

[0035] In some embodiments, the permission-based NFT remixing mechanism may also allow the creator of the input NFT to control ownership of the input NFT for the user who performs the NFT remixing, referred to herein as the NFT remixer. For example, in some cases, the creator may disallow the NFT remixer to maintain a copy of the input NFT after performing the NFT remixing, in which cases the mechanism may invalidate the input NFT associated with the account of the NFT remixer. In some cases, the creator may allow the NFT remixer to maintain a copy

of the input NFT after performing the NFT remixing, in which cases the mechanism may maintain existence of the input NFT associated with the account of the NFT remixer.

[0036] In some embodiments, a user, e.g., the current owner of two input NFTs of different types (e.g., an original song and an original video footage) may generate an output NFT based on the two input NFTs of different types (e.g., to generate a music video based on the original song and the original video footage). Accordingly, the permission-based NFT remixing mechanism may include checking whether the creators (e.g., a songwriter and a videographer) permit NFT remixing for different types of NFTs. The mechanism may also check whether the different types of NFTs share the same algorithm for the cross-type NFT remixing, as well as conform to the same conditions of performing the NFT remixing.

[0037] In some embodiments, a method for remixing NFTs may include obtaining a first NFT of a first type and a second NFT of a second type based on a request of a user. For example, the user may submit a request for remixing NFTs by using a device communicatively coupled to a blockchain. The first NFT may be stored on the blockchain and associated with a first identifier (e.g., an alphanumeric string) and first metadata. The second NFT may be stored on the blockchain and associated with a second identifier and second metadata. By way of example, the first metadata and the second metadata may be defined in a JSON format, such as the format as described in association with Table 1.

[0038] In some embodiments, the first type and the second type may be the same or different. By way of example, the first type may be stored as an indicator or attribute in the first NFT, and the second type may be stored as an indicator or attribute in the second NFT. As another example, the first type and the second type

may be stored in a data structure (e.g., a database entry) separate from the first NFT and the second NFT, in which the blockchain system may access the data structure to determine the first type for the first NFT and the second type for the second NFT. In another example, the first type or the second type may be recorded as a part of the first identifier or the second identifier, respectively (e.g., the first identifier “E000234” indicating that the first type is a type “E”).

[0039] In some embodiments, to obtain the first NFT and the second NFT, the method may include obtaining a first parameter and a second parameter from the user. The method may further include obtaining the first NFT and the second NFT based on the first parameter and the second parameter. In some embodiments, the first parameter may include the first identifier, and the second parameter may include the second identifier. By way of example, the first NFT and the second NFT may be stored on the blockchain. After being received, the first identifier and the second identifier may be used as search keys to locate the first NFT and the second NFT on the blockchain.

[0040] In some embodiments, the method for remixing NFTs may also include determining whether the user has remixing authorization of at least one of the first NFT or the second NFT. In some embodiments, determining whether the user has remixing authorization of at least one of the first NFT or the second NFT may include determining whether the user has at least one of: an ownership of the first NFT, an approval granted to the user by a first owner of the first NFT, an ownership of the second NFT, or an approval granted to the user by a second owner of the second NFT. For example, the first metadata may include a first indicator or attribute for storing data representing whether the user has the remixing authorization of the first NFT. The second data may also include a second indicator or attribute for storing

data representing whether the user has the remixing authorization of the second NFT. By way of example, to determine whether the user has remixing authorization of at least one of the first NFT or the second NFT, the computing device may retrieve the value of the first indicator and the second indicator from the first NFT and the second NFT, respectively, and determine whether the user has remixing authorization based on their values. In some embodiments, the user's remixing authorization of at least one of the first NFT or the second NFT may be stored in a data structure, e.g., a database entry, separate from the first NFT or the second NFT.

[0041] In some embodiments, if the user has the remixing authorization of at least one of the first NFT or the second NFT, the method for remixing NFTs may further include determining whether remixing is permissible for the first NFT and the second NFT based on the first NFT and the second NFT. In some embodiments, the first NFT may include a first indicator representing that the remixing is permissible for the first NFT, and the second NFT may include a second indicator representing that the remixing is permissible for the second NFT.

[0042] In some embodiments, to determine whether the remixing is permissible for the first NFT and the second NFT based on the first NFT and the second NFT, the method may include determining whether the first type is the same as the second type. If the first type is the same as the second type, the method may further include determining that the remixing is permissible for the first NFT and the remixing is permissible for the second NFT.

[0043] In some embodiments, to determine whether the remixing is permissible for the first NFT and the second NFT based on the first NFT and the second NFT, the method may include utilizing a smart contract, described below, to

determine whether the first type and the second type are compatible with an input type list, meaning that the smart contract supports remixing of the first type and the second type. If the first type and the second type are compatible with the input type list, the method may further include determining that the remixing is permissible for the first NFT and the second NFT.

[0044] By way of example, the smart contract may be provided by a service provider, and the smart contract is deployed on the blockchain where the first NFT and the second NFT are recorded. For example, in some embodiments, a provider who defines a particular type of NFT may provide a remixing function for that particular type of NFT. In another example, a first provider who defines a first type of NFT and a second provider who defines a second type of NFT may jointly provide a remixing function for remixing an NFT of the first type and an NFT of the second type. The smart contract may access an input type list. For example, the smart contract may define the input type list, e.g., as an attribute of the smart contract. As another example, the input type list may be stored in a data structure (e.g., as a database entry) accessible by the smart contract.

[0045] In the embodiments, when a type is compatible with the input type list, the type is permitted, allowed, or authorized to be remixed by the smart contract based on the input type list. For example, a type (e.g., the first type or the second type above) may be checked or looked up to determine whether it is included in the input type list. If it is included, the type is compatible with the input type list.

Otherwise, the type is not compatible with the input type list. In some embodiments, an administrator program (e.g., an application programming interface, a program process, or the like) may be utilized to manage the input type list. The administrator program may add, register, delete, or remove a type to or from the input type list.

[0046] In some embodiments, to determine whether the remixing is permissible for the first NFT and the second NFT based on the first NFT and the second NFT, the method may include utilizing the smart contract to determine whether the first type is compatible with the second type. If the first type is compatible with the second type, the method may further include determining that the remixing is permissible for the first NFT and the second NFT.

[0047] In the embodiments, when the first type is compatible with the second type, the first type and the second type are permitted, allowed, or authorized to be remixed by the smart contract. For example, the smart contract may define whether the first type may be remixed with the second type. If it is so defined, the first type is compatible with the second type. Otherwise, the first type is not compatible with the second type.

[0048] In some embodiments, if the remixing is permissible for the first NFT and the second NFT, the method may further include generating an output identifier based on the first identifier and the second identifier, generating an output type based on the first type and the second type, and generating output metadata based on the first metadata and the second metadata.

[0049] In some embodiments, to generate the output identifier based on the first identifier and the second identifier, the method may include generating a concatenation value by concatenating the first identifier and the second identifier. Then, the method may further include generating a hash value of the concatenation value as the output identifier. By doing so, the method may enable verifying whether the output identifier represents an output NFT remixed from the first NFT and the second NFT rather than other NFTs. Such verification may ensure authentication and security of the remixed NFTs.

[0050] By way of example, the computing device may generate the output identifier using Equation (1):

$$tid^{output} = H(tid_1^{input} || tid_2^{input} || \dots || tid_L^{input}) \quad \text{Eq. (1)}$$

[0051] In Eq. (1), tid^{output} may represent the output identifier, and $tid_1^{input}, tid_2^{input}, \dots, tid_L^{input}$ may represent L (L being an integer) input identifiers of L NFTs, respectively, used as inputs for NFT remixing, including the first identifier (e.g., tid_1^{input}) and the second identifier (e.g., tid_2^{input}). The operator $||$ may represent a concatenation operation. For example, concatenating the first identifier and the second identifier may be implemented as a string joining operation, in which the first identifier and the second identifier may be treated as two strings and joined by order (e.g., the second identifier following the first identifier) to form a new string. In some embodiments, $tid_1^{input} || tid_2^{input} || \dots || tid_L^{input}$ may represent an ordered (e.g., in an lexicographical order) concatenation of $tid_1^{input}, tid_2^{input}, \dots, tid_L^{input}$. In Eq. (1), the symbol $H()$ may represent a hash function.

[0052] In some embodiments, to generate the output metadata based on the first metadata and the second metadata, the method may include invoking a smart contract stored on the blockchain. The smart contract may include a remixing function. Then, the method may further include inputting the first metadata and the second metadata to the remixing function to generate the output metadata. In some embodiments, the remixing function may be provided by the user of the NFT remixing. In some embodiments, the remixing function may be provided by a provider other than the user. For example, in some embodiments, a provider who defined a particular type of NFT may provide a remixing function for that particular type of NFT. In another example, a first provider who defined a first type of NFT and

a second provider who defined a second type of NFT may jointly provide a remixing function for remixing the first type of NFT and the second type of NFT.

[0053] The smart contract may be a computer protocol 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. By way of example, users of the blockchain (e.g., 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. 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 include operational code that is fully or partially executed without human interaction.

[0054] By way of example, the remixing function included in the smart contract may be represented as Equation (2):

$$meta^{output} = F(\{meta_i^{input}\}_{i=1,...,L}) \quad \text{Eq. (2)}$$

[0055] In Eq. (2), $meta^{output}$ may represent the output metadata, and $\{meta_i^{input}\}_{i=1,...,L}$ may represent a set of L (L being an integer) entries of input metadata of L NFTs, respectively, used as inputs for NFT remixing, including the first metadata (e.g., $meta_1^{input}$) and the second metadata (e.g., $meta_2^{input}$). The symbol $F()$ represents the remixing function. In some embodiments, the first metadata may include a first value, and the second metadata may include a second value. As an example, the first value and the second value may be stored in fields representing attributes of the first NFT and the second NFT in JSON formats, e.g., the JSON format as described in association with Table 1.

[0056] In some embodiments, to input the first metadata and the second

metadata to the remixing function to generate the output metadata, the method may further include determining a value of the output metadata corresponding to the first value and the second value based on an average value of the first value and the second value. By way of example, with reference to Eq. (2), $F()$ may include an average function that receives the first value and the second value as inputs and output an average of the first value and the second value as a value to be included in the output metadata. As an example, the value to be included in the output metadata may be stored in a field representing an attribute of the output NFT in a JSON format, e.g., the same format as for the first NFT and the second NFT.

[0057] In some embodiments, to input the first metadata and the second metadata to the remixing function to generate the output metadata, the method may include obtaining a particular value, e.g., an alphanumeric value for representing a name of the output NFT, from the user. Then, the method may further include inputting the particular value to the remixing function to generate a value, e.g., an attribute representing the name of the output NFT, of the output metadata. By way of example, the remixing function included in the smart contract may be represented as Equation (3):

$$meta^{output} = F(\{meta_i^{input}\}_{i=1,...,L}, \{v_i\}_{i=1,...,L}) \quad \text{Eq. (3)}$$

[0058] Eq. (3) is similar to Eq. (2), in which $\{v_i\}_{i=1,...,L}$ may represent the particular values obtained from the user corresponding to L (L being an integer) entries of input metadata of L NFTs, respectively, used as inputs for NFT remixing. As an example, with reference to Eq. (3), $F()$ may include an average function that receives the first value and the second value as inputs and output an average of the first value and the second value. Then, $F()$ may generate a value to be included in the output metadata by taking as inputs a particular value obtained from the user

(e.g., v_1 or v_2) and the average of the first value and the second value.

[0059] In some embodiments, to input the first metadata and the second metadata to the remixing function to generate the output metadata, the method may further include obtaining a provably secure random number from the blockchain. The method may then include inputting the first value, the second value, and the provably secure random number to the remixing function to generate a value of the output metadata corresponding to the first value and the second value. The provably secure random number may be a random number generated with provable security. By way of example, the blockchain may be communicatively coupled to a random number generator, e.g., implemented as a service, such as a random ORACLE® service, that may generate provably secure random numbers. By including the provably secure random number, the output metadata may be determined with uniqueness, which may be preferred in some application scenarios, e.g., remixing NFTs representing in-game characters or items.

[0060] By way of example, the remixing function included in the smart contract may also be represented as Equation (4):

$$meta^{output} = F(\{meta_i^{input}\}_{i=1,...,L}, R) \quad \text{Eq. (4)}$$

[0061] Eq. (4) is similar to Eq. (2), in which R may represent the provably secure random number. As an example, with reference to Eq. (4), $F()$ may include an average function that receives the first value and the second value as inputs and outputs an average of the first value and the second value. Then, $F()$ may generate a value to be included in the output metadata by taking as inputs a provably secure random number (e.g., R) obtained from the blockchain, e.g., via a service communicatively coupled to blockchain system 100, and the average of the first value and the second value.

[0062] By way of example, the remixing function included in the smart contract may also be represented as Equation (5):

$$meta^{output} = F(\{meta_i^{input}\}_{i=1,...,L}, R, \{v_i\}_{i=1,...,L}) \quad \text{Eq. (5)}$$

[0063] Eq. (5) is similar to Eq. (4). As an example, with reference to Eq. (5), $F()$ may include an average function that receives the first value and the second value as inputs and output an average of the first value and the second value. Then, $F()$ may generate a value to be included in the output metadata by taking as inputs a provably secure random number (e.g., R) obtained from the blockchain, e.g., via a service communicatively coupled to blockchain system 100, a particular value obtained from the user (e.g., v_1 or v_2), and the average of the first value and the second value.

[0064] In some embodiments, the method for remixing NFTs may further include generating an output NFT based on the output identifier, the output type, and the output metadata, and recording the output NFT on the blockchain.

[0065] In some embodiments, if the user has the remixing authorization of at least one of the first NFT or the second NFT, the method for remixing NFTs may further include determining whether a first indicator represents that remixing is permissible for the first NFT and a second indicator represents that the remixing is permissible for the second NFT, in which the first NFT includes the first indicator, and the second NFT includes the second indicator. If the first indicator represents that the remixing is permissible for the first NFT and the second indicator represents that the remixing is permissible for the second NFT, the method may further include generating the output identifier based on the first identifier and the second identifier, and generating the output metadata based on the first metadata and the second metadata. An NFT (e.g., the first NFT or the second NFT) may include an indicator

(e.g., the first indicator or the second indicator) as data separate from its identifier (e.g., the first identifier or the second identifier) and its metadata (e.g., the first metadata or the second metadata), or include the indicator (e.g., the first indicator or the second indicator) in its metadata (e.g., the first metadata or the second metadata). By way of example, the first metadata may include the first indicator, and the second metadata may include the second indicator.

[0066] In some embodiments, the first indicator may include a first numeric value, and the second indicator may include a second numeric value. The first indicator may represent that the remixing is not permissible for the first NFT when the first numeric value is negative, and that the remixing is permissible for the first NFT when the first numeric value is positive or zero. The second indicator may represent that the remixing is not permissible for the second NFT when the second numeric value is negative, and that the remixing is permissible for the second NFT when the second numeric value is positive or zero.

[0067] By way of example, with reference to Table 1, each of the first metadata and the second metadata may be defined in accordance with the JSON format described in association with Table 1. Also, each of the first metadata and the second metadata may include an additional field, e.g., for storing an integer or a floating number, which may be the first indicator or the second indicator. A creator of the first NFT or the second NFT (e.g., a songwriter if the NFT represents a song, a photographer if the NFT represents a photo, a player if the NFT represents a game character, or the like) may have permission to control the value of the first indicator or the second indicator, respectively. If the value of the first indicator or the second indicator is set to be negative, it may represent that the creator disallows remixing of the first NFT or the second NFT. In such cases, the computing device may

terminate NFT remixing operations. If the value of the first indicator or the second indicator is set to be zero or positive, it may represent that the creator allows the remixing of the first NFT or the second NFT. In such cases, the computing device may continue the NFT remixing operations.

[0068] In some embodiments, the first indicator may further represent a first fee when the first numeric value is positive or zero, and the second indicator may further represent a second fee when the second numeric value is positive or zero. If the remixing is permissible for the first NFT and the second NFT, the method for remixing NFTs may further include sending data to the blockchain to enable deduction of at least one of the first fee or the second fee from an account associated with the user.

[0069] By way of example, with reference to Table 1, each of the first metadata and the second metadata may be defined in accordance with the JSON format described in association with Table 1. Also, each of the first metadata and the second metadata may include an additional field, e.g., for storing an integer or a floating number, which may be referred to as the first indicator or the second indicator. If the value of the first indicator or the second indicator is set to be zero or positive, it may represent that the creator allows the remixing of the first NFT or the second NFT, and the value of the first indicator or the second indicator represents the first fee and the second fee, respectively. For example, if the first NFT or the second NFT represents an artwork, the first fee or the second fee may represent a loyalty fee. In such cases, the computing device may send data to the blockchain to enable deduction of the loyalty fee from the account associated with the user to an account associated with the creator. For example, such deduction of fees may be conducted using a native token, e.g., a cryptocurrency, of the blockchain.

[0070] In some embodiments, if the user does not have ownership of at least one of the first NFT or the second NFT, the method for remixing NFTs may further include determining whether the user is authorized for the remixing by a first owner of the first NFT and a second owner of the second NFT. If the user is authorized for the remixing by the first owner and the second owner, the method may further include determining whether the first indicator represents that the remixing is permissible for the first NFT and the second indicator represents that the remixing is permissible for the second NFT.

[0071] By way of example, digital signatures may be used to determine whether the user is authorized for the remixing by the first owner and the second owner. For example, if the user does not have ownership of an NFT (e.g., the first NFT or the second NFT), an approval document may be sent to an owner of the NFT (e.g., the first owner or the second owner) for a digital signature (e.g., signing using the owner's private key). After the owner providing the digital signature, the signed approval document may be sent back to the blockchain. The blockchain may then verify the owner's signature using the owner's public key. If the verification is successful, the blockchain may deem the user is authorized for the remixing by the owner, and continue to process the user's request for remixing.

[0072] In another example, the user does not have ownership of an NFT (e.g., the first NFT or the second NFT) but may have prior authorization from an owner of the NFT (e.g., the first owner or the second owner) for the remixing. Such a prior authorization may be stored as an indicator (e.g., a third indicator in the first NFT, or a fourth indicator in the second NFT) in a data structure (e.g., an array). The data structure may be separate from the NFT. In such cases, when the user does not have ownership of the NFT (e.g., the first NFT or the second NFT), the blockchain

may retrieve the indicator (e.g., the third indicator or the fourth indicator) from the data structure and determine whether the indicator represents that the user is authorized for the remixing by the owner (e.g., the first owner or the second owner).

[0073] In some embodiments, besides the above-described operations, the method for remixing NFTs may further include providing a user interface (e.g., a graphic user interface, a command-line interface, an application programming interface, or the like) for obtaining the first NFT and the second NFT for the remixing. For example, a website graphic user interface (GUI) may be provided for the user to submit the first NFT and the second NFT for the remixing.

[0074] In some embodiments, in response to generating the output NFT on the blockchain, the method for remixing NFTs may further include invalidating at least one of the first NFT or the second NFT on the blockchain. Invalidating a token on a blockchain may include an operation of marking the token as unavailable (or spent) for any future transaction on the blockchain. For example, the blockchain may implement a spent pool to identify invalidated tokens (e.g., NFTs), and invalidating a token may include adding such a token into the spent pool. For any future transaction involving such a token, a transaction process may include determining whether the token is in the spent pool. If so, the transaction may be terminated.

[0075] In some embodiments, the first NFT may be of a first type, and the second NFT may be of a second type. The first type and the second type may be the same or different. By way of example, the first type may be stored as an indicator or attribute in the first NFT, and the second type may be stored as an indicator or attribute in the second NFT. As another example, the first type and the second type may be stored in a data structure, e.g., a database entry, separate from

the first NFT and the second NFT, in which the blockchain system may access the data structure to determine the first type for the first NFT and the second type for the second NFT. In another example, the first type or the second type may be recorded as a part of the first identifier or the second identifier, respectively, e.g., the first identifier “E000234” indicating that the first type is a type “E”. In some embodiments, if the user has the remixing authorization of at least one of the first NFT or the second NFT, the method for remixing NFTs may further include generating an output type for the output NFT based on the first type and the second type.

[0076] In some embodiments, to generate the output type based on the first type and the second type, the method may include invoking a smart contract. The smart contract may include a mapping relationship that maps the first type and the second type to the output type. The method may further include determining the output type based on the mapping relationship. For example, the smart contract may use an interface standard, e.g., an ERC-20 standard, an ERC-721 standard, or the like, for remixing NFTs.

[0077] By way of example, the blockchain system 100, through one or more smart contracts executing on the blockchain system 100, may include a mapping relationship that maps the first type and the second type to the output type, which is represented as Equation (6):

$$(TYPE_1, TYPE_2, \dots, TYPE_L) \rightarrow TYPE_O \quad \text{Eq. (6)}$$

[0078] In Eq. (6), $TYPE_O$ represents the output type, and $TYPE_1, TYPE_2, \dots, TYPE_L$ may represent L (L being an integer) input types of L NFTs, respectively, used as inputs for NFT remixing, including the first type (e.g., $TYPE_1$) and the second type (e.g., $TYPE_2$). For example, the mapping relationship represented by Eq. (6) may be implemented as a mapping table. To generate the

output type based on the mapping relationship, the computing device may use the first type and the second type as search keys to look up the mapping table, locate a candidate type corresponding to the first type and the second type, and determine the candidate type as the output type. In some embodiments, if the computing device cannot locate any candidate type corresponding to the first type and the second type after looking up the mapping table, the computing device may terminate the NFT remixing operations.

[0079] In Eq. (6), $TYPE_1, TYPE_2, \dots, TYPE_L$ may be all the same type or different types. For example, if the first type and the second type are a type representing songs, the output type may also be the type representing songs. As another example, if the first type and the second type are a type representing a photo and a type representing a song, respectively, the output type may be a type representing a video. It should be noted that the mapping between the input types and the output type of the mapping relationship may be various and not limited to the examples described herein.

[0080] In some embodiments, the remixing function (e.g., $F()$ described in association with Eqs. (2)-(5)) may be different if the mapping relationship (e.g., the mapping relationship described in association with Eq. (6)) that maps input NFT types to output NFT types are different. For example, for a mapping relationship $(TYPE_1, TYPE_2) \rightarrow TYPE_O$ (e.g., the first NFT and the second NFT being of $TYPE_1$ and $TYPE_2$, respectively), the remixing function may be

$F_1(\{meta_i^{input}\}_{i=1,2}, R, \{v_i\}_{i=1,2})$. As another example, for a mapping relationship

$(TYPE_1, TYPE_1) \rightarrow TYPE_O$ (e.g., the first NFT and the second NFT both being of

$TYPE_1$), the remixing function may be $F_2(\{meta_i^{input}\}_{i=1,2}, R, \{v_i\}_{i=1,2})$ that is different

from $F_1(\{meta_i^{input}\}_{i=1,2}, R, \{v_i\}_{i=1,2})$.

[0081] By way of example, for an NFT of a SUPER type representing a collectible, Table 2 presents an example JSON object for the metadata of the NFT, which includes multiple attributes (e.g., a name of the collectible, a public key address of a creator of the collectible, and a score that determines the value of the collectible) of the SUPER-type NFT.

Line No.	
1	{
2	"title": "Metadata of SUPER NFT",
3	"type": "object",
4	"properties": {
5	"name": {
66	"type": "string",
7	"description": "collectible name"
8	},
9	"creator": {
10	"type": "string",
11	"description": "public key address of the creator"
12	},
13	"score": {
14	"type": "short",
15	"description": "a score of the NFT determines its value"
16	},
17	}
18	}

Table 2 Example Metadata of SUPER-Type NFT

[0082] Assuming that a blockchain system (e.g., blockchain system 100) includes a smart contract defining a remixing function and that the remixing function includes a mapping relationship $(SUPER, SUPER) \rightarrow SUPER$ (e.g., mapping two NFTs of SUPER type to an output NFT of SUPER type), Table 3 presents example pseudocode of the smart contract.

Line No.	
----------	--

1	Smart contract Remix1 {
2	ENUM[] input_type_list =[SUPER, SUPER];
3	ENUM output_type = SUPER;
4	remix(NFT t1, NFT t2, String name) {
5	ABORT if t1.type != input_type_list[1] or t1.owner != user
6	ABORT if t2.type != input_type_list[2] or t2.owner != user
7	ABORT if t1.remix_fee == 0 or t2.remix_fee == 0
8	accounts[user] -= t1.remix_fee + t2.remix_fee
9	accounts[t1.creator] += t1.remix_fee
10	accounts[t2.creator] += t2.remix_fee
11	initialize t3 with tid=H(t1.tid t2.tid)
12	//determine metadata of t3
13	t3.name = name
14	t3.creator = user
15	fetch R from random database service
16	t3.score = (t1.score+t2.score)/2*normalize(R,[0,1])
17	invalidate t1 and t2, save t3 to the ledger.
18	}

Table 3 Example Pseudocode of Smart Contract for remixing SUPER-Type NFTs

[0083] As illustrated in Table 3, the SUPER type is implemented as an enumeration type (“ENUM”). The smart contract includes an input type list “input_type_list” that defines permitted types for NFT remixing. The smart contract also includes a remixing function “remix()” that receives a first NFT “t1,” a second NFT “t2,” and a string “name” that represents the name of the output NFT “t3.”

[0084] As illustrated in Table 3, the remixing function may check whether t1 and t2 are of the type listed in the input type list “input_type_list” and whether a user (e.g., a debtor) has remixing authorization (e.g., ownership) of t1 and t2. If at least one of t1 or t2 are of a type not listed in “input_type_list,” or if the user has no ownership of at least one of t1 and t2, the NFT remixing will be aborted. Each of t1 and t2 includes an indicator “remix_fee” for the remixing function to determine whether the user has permission to remix t1 and t2, in which a positive value of “remix_fee” represents that the user has the permission, and a zero value of

“remix_fee” represents that the user has no permission. The positive value of “remix_fee” also represents the collectible fees owed by the user to a creator (e.g., a creditor) of the NFT.

[0085] If “remix_fee” is positive for both t1 and t2, the remixing function may add the “remix_fee” value of t1 and the “remix_fee” value of t2 to determine a sum, treat the sum as the fee to be collected from the user, and deduct the sum from an account of the user. The remixing function may further add the “remix_fee” value of t1 to an account of a creator (e.g., a first creditor) of t1, and add the “remix_fee” value of t2 (e.g., a second creditor) to an account of a creator of t2.

[0086] The remixing function may further concatenate the identifiers (represented as “tid”) of t1 and t2, and determine an identifier of t3 by hashing the concatenation. Moreover, the remixing function may determine the name of t3 as the inputted string “name” and determine the creator of t3 as the user. After obtaining a provably secure random number “R” from a database service, the remixing function may use “R” to determine a value “score” for t3, which represents the collectible value of t3. As illustrated, the value “score” of t3 may be generated using the values of “score” of t1 and t2 as well as “R” in accordance with $t3.score = (t1.score + t2.score) / 2 * \text{normalize}(R, [0, 1])$. After that, the remixing function may invalidate t1 and t2 from the blockchain, and save t3 to the blockchain.

[0087] In some embodiments, in response to generating the output NFT on the blockchain, the method for remixing NFTs may also include maintaining at least one of the first NFT or the second NFT on the blockchain. For example, the computing device may maintain existence of at least one of the first NFT or the second NFT on the blockchain.

[0088] By way of example, with reference to Table 2, an NFT of a SUPER

type representing a collectible may have its metadata defined in a format as described in association with Table 2. Assuming that a blockchain system (e.g., blockchain system 100) includes a smart contract defining a remixing function and that the remixing function includes a mapping relationship $(\text{SUPER}, \text{SUPER}) \rightarrow \text{SUPER}$ (e.g., mapping two NFTs of SUPER type to an output NFT of SUPER type), Table 4 presents example pseudocode of the smart contract that maintains at least one of the first NFT or the second NFT on the blockchain after the NFT remixing.

Line No.	
1	Smart contract Remix2 {
2	ENUM[] input_type_list =[SUPER, SUPER];
3	ENUM output_type = SUPER;
4	approve(NFT t, uint fee) {
5	ABORT if t.type not in input_type_list
6	ABORT if t.owner != user
7	Save approvals[t.tid]=fee
8	}
9	remix(NFT t1, NFT t2, String name) {
10	ABORT if t1.type != input_type_list[1]
11	ABORT if t2.type != input_type_list[2]
12	ABORT if t1.remix_fee == 0 or t2.remix_fee == 0
13	accounts[user] -= t1.remix_fee + t2.remix_fee
14	accounts[t1.creator] += t1.remix_fee
15	accounts[t2.creator] += t2.remix_fee
16	ABORT if approvals[t1.tid] not exist or approvals[t2.tid] not exist
17	accounts[user] -= approvals[t1.tid] + approvals[t2.tid]
18	accounts[t1.owner] += approvals[t1.tid]
19	accounts[t2.owner] += approvals[t2.tid]
20	initialize t3 with tid=H(t1.tid t2.tid)
21	// determine metadata of t3
22	t3.name = name
23	t3.creator = user
24	fetch R from random oracle service
25	t3.score = (t1.score+t2.score)/2*normalize(R,[0,1])
26	save t3 to the ledger
27	}

Table 4 Example Pseudocode of Smart Contract for remixing SUPER-Type NFTs

[0089] The smart contract illustrated in Table 4 may be similar to the smart contract illustrated in Table 3. As illustrated in Table 4, the SUPER type is implemented as an enumeration type (“ENUM”). The smart contract includes an input type list “input_type_list” that defines permitted types for NFT remixing. The smart contract also includes a remixing function “remix()” that receives a first NFT “t1,” a second NFT “t2,” and a string “name” that represents the name of the output NFT “t3.” Compared with the smart contract illustrated in Table 3, the smart contract illustrated in Table 4 includes a function “approve()” for checking approval of NFT remixing granted to a user (e.g., a debtor) of “t” by an owner (e.g., an original creditor) of “t”. The “approve()” function may receive an NFT “t” and an integer “fee” (representing a collectible value owed to the owner) as inputs. If a type of “t” is in the input type list “input_type_list” and the owner of “t” is not the user, the “approve()” function may save the value of “fee” into an array “approvals” using an identifier of “t” (i.e., “t.tid”) as an index. For example, the “approve()” function may be executed for both “t1” and “t2” by receiving “t1” and “fee” as input parameters, and receiving “t2” and “fee” as input parameters, respectively.

[0090] As illustrated in Table 4, the remixing function may check whether t1 and t2 are of the type listed in the input type list “input_type_list.” If at least one of t1 or t2 are of a type not listed in “input_type_list,” the NFT remixing will be aborted. Each of t1 and t2 includes an indicator “remix_fee” for the remixing function to determine whether the user has permission to remix t1 and t2, in which a positive value of “remix_fee” represents that the user has the permission, and a zero value of “remix_fee” represents that the user has no permission. The positive value of “remix_fee” also represents the collectible fees owed by the user to a creator (e.g., a sub-loaner) of the NFT.

[0091] If “remix_fee” is positive for both t1 and t2, the remixing function may add the “remix_fee” value of t1 and the “remix_fee” value of t2 to determine a sum, treat the sum as the fee to be collected from the user, and deduct the sum from an account of the user. The remixing function may further add the “remix_fee” value of t1 to an account of a creator (e.g., a first sub-loaner) of t1, and add the “remix_fee” value of t2 to an account of a creator (e.g., a second sub-loaner) of t2.

[0092] Then, the remixing function may check whether the user as a non-owner of t1 and t2 has approval to perform the NFT remixing. The remixing function may check whether both “approvals[t1.tid]” and “approvals[t2.tid]” exist. If both “approvals[t1.tid]” and “approvals[t2.tid]” exist, it represents that the user has approval from t1’s owner (e.g., a first original creditor that loans to the first sub-loaner) and approval from t2’s owner (e.g., a second original creditor that loans to the second sub-loaner) for performing the NFT remixing. Otherwise, the NFT remixing may be aborted. The values of “approvals[t1.tid]” and “approvals[t2.tid]” may represent collectible fees owed by the user to the first owner and the second owner, respectively.

[0093] If both “approvals[t1.tid]” and “approvals[t2.tid]” exist, the remixing function may add the value of “approvals[t1.tid]” and the value of “approvals[t2.tid]” to determine a sum, treat the sum as the fee to be collected from the user, and deduct the sum from the account of the user. The remixing function may further add the value of “approvals[t1.tid]” value to an account of t1’s owner, and add the value of “approvals[t2.tid]” to an account of t2’s owner.

[0094] The remixing function of Table 4 may further determine an identifier of t3, the name of t3, the creator of t3, and the value “score” of t3 in the same way as described in association with Table 3. After that, the remixing function may maintain

t1 and t2 on the blockchain, and save t3 to the blockchain.

[0095] In some embodiments, the first type of the first NFT may be the same as the second type of the second NFT. For example, a blockchain may be used by a service provider, and the service provider may provide its users to generate and control NFTs of the same type. In some embodiments, the first type may be different from the second type. For example, a blockchain may be used by multiple service providers, such as a first service (e.g., game) provider and a second service (e.g., game) provider. The first service provider may provide its users to generate and control NFTs of a first type, and the second service provider may provide its users to generate and control NFTs of a second type.

[0096] FIG. 3 illustrates a flow chart of a method 300 for remixing NFTs, according to an embodiment. The method 300 may be performed by a computer, e.g., the computing device 200 (FIG. 2), coupled to 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 computer may or may not be a node of the blockchain system.

[0097] At step 302, the computer communicatively coupled to the blockchain system may obtain a first NFT (e.g., “t1” in Table 3 or Table 4) and a second NFT (e.g., “t2” in Table 3 or Table 4) based on a request of a user. The first NFT may be stored on a blockchain and associated with a first identifier (e.g., “t1.tid” in Table 3 or Table 4) and first metadata (e.g., including “t1.creator” and “t1.score” in Table 3 or Table 4). The second NFT may be stored on the blockchain and associated with a second identifier (e.g., “t2.tid” in Table 3 or Table 4), and second metadata (e.g., including “t2.creator” and “t1.score” in Table 3 or Table 4). In some embodiments,

the first NFT may be of a first type (e.g., “t1.type” in Table 3 or Table 4), and the second NFT may be of a second type (e.g., “t2.type” in Table 3 or Table 4). The first type and the second type may be the same or different.

[0098] In some embodiments, to obtain the first NFT and the second NFT, the computer may obtain a first parameter and a second parameter from the user. The computer may further obtain the first NFT and the second NFT based on the first parameter and the second parameter. In some embodiments, the first parameter may include the first identifier, and the second parameter may include the second identifier.

[0099] At step 304, the computer may determine whether the user has remixing authorization of at least one of the first NFT or the second NFT. In some embodiments, determining whether the user has remixing authorization of at least one of the first NFT or the second NFT may include determining whether the user has at least one of: an ownership of the first NFT, an approval granted to the user by a first owner of the first NFT, an ownership of the second NFT, or an approval granted to the user by a second owner of the second NFT.

[0100] At step 306, if the user has the remixing authorization of at least one of the first NFT or the second NFT, the computer may determine whether remixing is permissible for the first NFT and the second NFT based on the first NFT and the second NFT, as described above. In some embodiments, the first NFT may include a first indicator (e.g., “t1.remix_fee” in Table 3 or Table 4) represents that the remixing is permissible for the first NFT, and the second NFT may include a second indicator (e.g., “t2.remix_fee” in Table 3 or Table 4) represents that the remixing is permissible for the second NFT.

[0101] At step 308, if the remixing is permissible for the first NFT and the

second NFT, the computer may generate an output identifier (e.g., “t3.tid” in Table 3 or Table 4) based on the first identifier and the second identifier, and generate an output type (e.g., “SUPER” in Table 3 or Table 4) based on the first type “SUPER” and the second type “SUPER”), generate output metadata (e.g., including “t3.name,” “t3.creator,” and “t3.score” in Table 3 or Table 4) based on the first metadata and the second metadata.

[0102] In some embodiments, to generate the output identifier, the computer may generate a concatenation value (e.g., “t1.tid||t2.tid” in Table 3 or Table 4) by concatenating the first identifier and the second identifier, then generate the output identifier as a hash value (e.g., “H(t1.tid||t2.tid)” in Table 3 or Table 4) of the concatenation value.

[0103] In some embodiments, to generate the output metadata, the computer may invoke a smart contract. The smart contract may include a remixing function. Then, the computer may input the first metadata and the second metadata to the remixing function to generate the output metadata. In some embodiments, the first metadata may include a first value, and the second metadata may include a second value.

[0104] For example, to generate the output metadata, the computer may obtain a particular value (e.g., “name” in Table 3 or Table 4) from the user, and input the particular value to the remixing function to generate a value (e.g., “t3.name” in Table 3 or Table 4) of the output metadata. As another example, to generate the output metadata, the computer may obtain a provably secure random number (e.g., “R” in Table 3 or Table 4) from the blockchain, and input the first value, the second value, and the provably secure random number to the remixing function to generate a value of the output metadata corresponding to the first value and the second value.

[0105] Still referring to FIG. 3, at step 310, the computer may generate an output NFT based on the output identifier, the output type, and the output metadata, and record the output NFT on the blockchain. In some embodiments, if a smart contract is utilized to determine whether the first type and the second type are compatible with the input type list at step 306, at step 310, the computer may utilize the smart contract to determine the output type based on the first type and the second type. For example, the smart contract may include or access a mapping relationship (e.g., the mapping relationship described in association with Eq. (6)). The mapping relationship may map the first type and the second type to the output type. Then, the computer may generate the output type based on the mapping relationship.

[0106] In some embodiments, in response to generating the output NFT on the blockchain, the computer may invalidate at least one of the first NFT or the second NFT on the blockchain. In some embodiments, in response to generating the output NFT on the blockchain, the computer may maintain at least one of the first NFT or the second NFT on the blockchain.

[0107] In some embodiments, besides steps 302-310, if the user has the remixing authorization of at least one of the first NFT or the second NFT, the computer may determine whether a first indicator (e.g., "t1.remix_fee" in Table 3 or Table 4) represents that remixing is permissible for the first NFT and a second indicator (e.g., "t2.remix_fee" in Table 3 or Table 4) represents that the remixing is permissible for the second NFT. The first NFT may include the first indicator, and the second NFT may include the second indicator. If the first indicator represents that the remixing is permissible for the first NFT and the second indicator represents that the remixing is permissible for the second NFT, the computer may generate the

output identifier based on the first identifier and the second identifier, generate the output type based on the first type and the second type, and generate the output metadata based on the first metadata and the second metadata.

[0108] In some embodiments, the first indicator may have a first numeric value, and the second indicator may have a second numeric value. The first indicator may represent that the remixing is not permissible for the first NFT when the first numeric value is negative and is permissible for the first NFT when the first numeric value is positive or zero. The second indicator may represent that the remixing is not permissible for the second NFT when the second numeric value is negative and is permissible for the second NFT when the second numeric value is positive or zero.

[0109] In some embodiments, the first indicator may further represent a first fee when the first numeric value is positive or zero, and the second indicator may further represent a second fee when the second numeric value is positive or zero. If the remixing is permissible for the first NFT and the second NFT, the computer may send data to the blockchain to enable deduction of at least one of the first fee or the second fee from an account associated with the user.

[0110] In some embodiments, if the user does not have ownership of at least one of the first NFT or the second NFT, the computer may determine whether the user is authorized for the remixing by a first owner (e.g., "t1.owner" in Table 4) of the first NFT and a second owner (e.g., "t2.owner" in Table 4) of the second NFT. If the user is authorized for the remixing by the first owner and the second owner, the computer may determine whether the first indicator represents that the remixing is permissible for the first NFT and the second indicator represents that the remixing is permissible for the second NFT.

[0111] By way of example, an array (e.g., “approval[]” in Table 4) may store a third indicator (e.g., “approval[t1.tid]” in Table 4) representing that the user is authorized for the remixing by the first owner and a fourth indicator (e.g., “approval[t2.tid]” in Table 4) representing that the user is authorized for the remixing by the second owner. If the user does not have the remixing authorization of at least one of the first NFT or the second NFT, the computer may retrieve the third indicator and the fourth indicator from the array, and determine whether the user is authorized for the remixing by the first owner and the second owner based on the third indicator and the fourth indicator.

[0112] In some embodiments, besides steps 302-310, the computer may further provide a user interface for the user to send the first NFT and the second NFT for the remixing.

[0113] FIG. 4 is a block diagram of an apparatus 400 for remixing NFTs, according to an embodiment. The apparatus 400 may be an implementation of a software process, a hardware circuit, or a combination thereof, and may correspond to the method 300 (FIG. 3). Referring to FIG. 4, the apparatus 400 may include an obtaining module 402, a remixing authorization determination module 404, a permission determination module 406, a remixing module 408, and a generation module 410.

[0114] The obtaining module 402 may obtain (e.g., by performing step 302) a first NFT of a first type and a second NFT of a second type, the first NFT being stored on a blockchain and associated with a first identifier and first metadata, and the second NFT being stored on the blockchain and associated with a second identifier and second metadata. The remixing authorization determination module 404 may determine (e.g., by performing step 304) whether the user has remixing

authorization of at least one of the first NFT or the second NFT. If the user has the remixing authorization of at least one of the first NFT or the second NFT, the permission determination module 406 may determine (e.g., by performing step 306) whether remixing is permissible for the first NFT and the second NFT based on the first NFT and the second NFT. If the remixing is permissible for the first NFT and the second NFT, the remixing module 408 may generate (e.g., by performing step 308) an output identifier based on the first identifier and the second identifier, generate an output type based on the first type and the second type, and generate output metadata based on the first metadata and the second metadata. The generation module 410 may generate (e.g., by performing step 310) an output NFT based on the output identifier, the output type, and the output metadata, and record the output NFT on the blockchain.

[0115] 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 400 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.

[0116] For an implementation process of functions and roles of each module in the apparatus 400, references can be made to corresponding steps in the above-described methods. Details are omitted here for simplicity.

[0117] 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.

[0118] 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.

[0119] 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).

[0120] 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.

[0121] 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.

[0122] 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.

[0123] 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 remixing non-fungible tokens (NFTs), the method comprising:

obtaining a first NFT of a first type and a second NFT of a second type based on a request of a user, the first NFT being stored on a blockchain and associated with a first identifier and first metadata, and the second NFT being stored on the blockchain and associated with a second identifier and second metadata;

determining whether the user has remixing authorization of at least one of the first NFT or the second NFT;

if the user has the remixing authorization of at least one of the first NFT or the second NFT, determining whether remixing is permissible for the first NFT and the second NFT based on the first NFT and the second NFT;

if the remixing is permissible for the first NFT and the second NFT, generating an output identifier based on the first identifier and the second identifier, generating an output type based on the first type and the second type, and generating output metadata based on the first metadata and the second metadata; and

generating an output NFT based on the output identifier, the output type, and the output metadata, and recording the output NFT on the blockchain.

2. The method of claim 1, wherein obtaining the first NFT of the first type and the second NFT of the second type comprises:

obtaining a first parameter and a second parameter from the user; and

obtaining the first NFT and the second NFT based on the first parameter and the second parameter.

3. The method of claim 2, wherein the first parameter comprises the first identifier, and the second parameter comprises the second identifier.
4. The method of claim 1, wherein determining whether the user has remixing authorization of at least one of the first NFT or the second NFT comprises:
determining whether the user has at least one of: an ownership of the first NFT, an approval granted to the user by a first owner of the first NFT, an ownership of the second NFT, or an approval granted to the user by a second owner of the second NFT.
5. The method of claim 1, wherein the first NFT comprises a first indicator representing that the remixing is permissible for the first NFT, and the second NFT comprises a second indicator representing that the remixing is permissible for the second NFT.
6. The method of claim 5, wherein the first indicator has a first numeric value, and the second indicator has a second numeric value,

the first indicator represents that the remixing is not permissible for the first NFT when the first numeric value is negative, and that the remixing is permissible for the first NFT when the first numeric value is positive or zero; and

the second indicator represents that the remixing is not permissible for the second NFT when the second numeric value is negative, and that the remixing is permissible for the second NFT when the second numeric value is positive or zero.

7. The method of claim 6, wherein the first indicator further represents a first fee when the first numeric value is positive or zero, and the second indicator further represents a second fee when the second numeric value is positive or zero, and the method further comprises:

if the remixing is permissible for the first NFT and the second NFT, sending data to the blockchain to enable deduction of at least one of the first fee or the second fee from an account associated with the user.

8. The method of claim 1, wherein determining whether the remixing is permissible for the first NFT and the second NFT based on the first NFT and the second NFT comprises:

determining whether the first type is the same as the second type; and

if the first type is the same as the second type, determining that the remixing is permissible for the first NFT and the second NFT.

9. The method of claim 1, wherein determining whether the remixing is permissible for the first NFT and the second NFT based on the first NFT and the second NFT comprises:

utilizing a smart contract to determine whether the first type and the second type are compatible with an input type list; and

if the first type and the second type are compatible with the input type list, determining that the remixing is permissible for the first NFT and the second NFT.

10. The method of claim 9, further comprising:

utilizing the smart contract to determine whether the first type is compatible with the second type; and

if the first type is compatible with the second type, determining that the remixing is permissible for the first NFT and the second NFT.

11. The method of claim 9, further comprising:

utilizing the smart contract to determine the output type based on the first type and the second type.

12. The method of claim 1, wherein generating the output identifier based on the first identifier and the second identifier comprises:

generating a concatenation value by concatenating the first identifier and the second identifier; and

generating a hash value of the concatenation value as the output identifier.

13. The method of claim 1, wherein generating the output metadata based on the first metadata and the second metadata comprises:

invoking a smart contract comprising a remixing function associated with the first type and the second type; and

inputting the first metadata and the second metadata to the remixing function to generate the output metadata.

14. The method of claim 13, wherein inputting the first metadata and the second metadata to the remixing function to generate the output metadata comprises:

obtaining a particular value from the user; and

inputting the particular value to the remixing function to generate a value of the output metadata.

15. The method of claim 13, wherein the first metadata comprises a first value, the second metadata comprises a second value, and inputting the first metadata and the second metadata to the remixing function to generate the output metadata comprises:

obtaining a provably secure random number from the blockchain; and
inputting the first value, the second value, and the provably secure random number to the remixing function to generate a value of the output metadata corresponding to the first value and the second value.

16. The method of claim 1, further comprising:

in response to generating the output NFT on the blockchain, invalidating or maintaining at least one of the first NFT or the second NFT on the blockchain.

17. The method of claim 1, wherein generating the output type based on the first type and the second type comprises:

invoking a smart contract comprising a mapping relationship, the mapping relationship mapping the first type and the second type to the output type; and
generating the output type based on the mapping relationship.

18. A device for processing a token, 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 17.

19. An apparatus for processing a token, the apparatus comprising a plurality of modules for performing the method of any of claims 1 to 17.

20. 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 17.