

METHODS AND DEVICES FOR PROTECTING AND VERIFYING STATE INFORMATION OF RECORD

TECHNICAL FIELD

[0001] The specification relates generally to computer technologies, and more particularly, to methods and devices for protecting and verifying state information of a record.

BACKGROUND

[0002] 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.

[0003] 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.

[0004] Users of a blockchain system may utilize the blockchain system to carry out various types of transactions. For example, two parties, Party A and Party B, may both own assets recorded on a blockchain maintained in the blockchain system. Parties A and B may enter into an agreement to exchange, or swap, their assets on the blockchain. Such agreements are typically carried out using smart contracts, which are computer protocols implemented in the form of computer code that are incorporated into the blockchain, to facilitate, verify, or enforce the negotiation or performance of contracts. In another example, Party A may submit a document, e.g., an invoice, to the blockchain and request Party B to review or endorse the document on the blockchain.

[0005] Some blockchain applications may model assets or documents recorded therein using various states, which may transition based on operations performed with respect to the recorded assets or documents. For example, a newly submitted document may be assigned a “submitted” state, which may transition to an “endorsed” state when the document becomes endorsed. The owner of the document with an “endorsed” state may use the document to accomplish certain tasks that may not be accomplished using a document with merely a “submitted” state. For example, suppose that the document is an invoice and the owner of the invoice wishes to use the invoice to secure financing (e.g., to borrow money against accounts receivable), a financial institution may refuse to approve the invoice for financing purposes if the invoice is merely in a “submitted” state. For example, the financial institution may impose a state transition rule that prohibits a transition directly from a “submitted” state to an “approved” state to prevent the owner of the document from bypassing the endorsement process.

[0006] Typically, contents of assets and documents recorded on blockchains

are encrypted to protect privacy. However, blockchain applications that model assets or documents recorded therein using various states typically record information concerning the states and operations associated with such states in plaintext.

[0007] Exposing information concerning the states and operations in this manner may provide opportunities for malicious parties to gain insights into the assets and documents recorded on the blockchains. For example, if a malicious party can learn from a blockchain that a document recorded on the blockchain has been through a “submitted” state and an “endorsed” state, and is now “under review” by a financial institution, then the malicious party may deduce that the document likely contains a description of an asset or an invoice that can be used to secure financing.

SUMMARY

[0008] In one aspect, a computer-implemented method for verifying state transition of a record includes: receiving a transaction submitted by a user, the transaction comprising an identification of the record, a protected state value representing a state of the record after performing an operation on the record, and a proof prepared by the user; determining whether the proof is acceptable by determining whether the proof indicates that the user is in possession of information generated at least partially based on a plaintext value of the state of the record after performing the operation; in response to a determination that the proof is unacceptable, refusing to process the transaction submitted by the user; and in response to a determination that the proof is acceptable, setting a recorded state of the record to the protected state value.

[0009] In another aspect, a device for verifying state transition of a record 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: receive a transaction submitted by a user, the transaction comprising an identification of the record, a protected state value representing a state of the record after performing an operation on the record, and a proof prepared by the user; determine whether the proof is acceptable by determining whether the proof indicates that the user is in possession of information generated at least partially based on a plaintext value of the state of the record after performing the operation; in response to a determination that the proof is unacceptable, refuse to process the transaction submitted by the user; and in response to a determination that the proof is acceptable, set a recorded state of the record to the protected state value.

[0010] 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 providing decentralized identity verification. The method includes: receiving a transaction submitted by a user, the transaction comprising an identification of the record, a protected state value representing a state of the record after performing an operation on the record, and a proof prepared by the user; determining whether the proof is acceptable by determining whether the proof indicates that the user is in possession of information generated at least partially based on a plaintext value of the state of the record after performing the operation; in response to a determination that the proof is unacceptable, refusing to process the transaction submitted by the user; and in response to a determination that the proof is acceptable, setting a recorded state of the record to the protected state value.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] 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.

[0012] FIG. 1 is a schematic diagram of a blockchain system, according to an embodiment.

[0013] FIG. 2 is a schematic diagram of a computing device for implementing a node in a blockchain system, according to an embodiment.

[0014] FIG. 3 is an illustration depicting a sample state diagram according to an embodiment.

[0015] FIG. 4 is a flow chart of a method for protecting and verifying state information of a record, according an embodiment.

[0016] FIG. 5 is a flow chart of a method for protecting and verifying state information of a record, according an embodiment.

[0017] FIG. 6 is a block diagram of an apparatus for protecting and verifying state information of a record, according to an embodiment.

DETAILED DESCRIPTION

[0018] Embodiments of the specification provide methods and devices for protecting and verifying state information of records (e.g., recorded assets or documents). The methods and devices may utilize encryption schemes or commitment schemes to protect information concerning states of recorded records. The methods and devices may also utilize blockchain systems to verify validities of the information concerning the states of the recorded records. Invalid state information

can be detected and fraudulent attempts to bypass certain states can be prevented. In this manner, the methods and devices can ensure that state information is protected while simultaneously enforcing state transition rules imposed on the states.

[0019] Embodiments disclosed in the specification have one or more technical effects. In some embodiments, the methods and devices permit users to protect information concerning states of recorded records. This allows the methods and devices to preserve privacy of such information. In some embodiments, the methods and devices require users to submit proofs in order to validate the information concerning the states of the recorded records. This allows the methods and devices to detect invalid state transitions and prevent fraudulent attempts to bypass certain states. In some embodiments, the methods and devices support validation of state information using a blockchain system. This allows the methods and devices to store state information in a data structure that can prevent tampering and manipulation by malicious parties. This also allows the methods and devices to utilize the blockchain system to validate state transitions in a public manner without revealing any private information to the public.

[0020] 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.

[0021] 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.

[0022] 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).

[0023] 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).

[0024] 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.

[0025] 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.

[0026] 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.

[0027] 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.

[0028] 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.

[0029] 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.

[0030] 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.

[0031] 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.

[0032] Users of a blockchain system, e.g., the blockchain system 100, may utilize the blockchain system 100 to carry out various types of transactions. For example, users may own assets recorded on a blockchain, e.g., the blockchain 120, maintained in the blockchain system 100, and enter into agreements to exchange, or swap, their assets on the blockchain 120. Users may also create and submit documents to the blockchain 120 for recordation and utilize the recorded documents for various purposes, including, e.g., invoice financing and the like.

[0033] In some embodiments, the blockchain 120 may model assets or documents recorded therein using various states, which may transition from one to another based on the operations performed on the assets or documents. For example, the blockchain 120 may enforce a set of state transition rules defined for managing the state of a record (e.g., an asset or a document) recorded on the blockchain 120 if the user who created the record wants to issue the record. FIG. 3 illustrates a sample state diagram 300 depicting a process for issuing a record. It is to be understood that the state diagram 300 is merely provided as an example and is not meant to be limiting.

[0034] As shown in FIG. 3, the state diagram 300 may include, e.g., five states S0-S4 and four operations F0-F3. The state diagram 300 also defines transitions, which specify how certain operations can trigger certain transitions from one state to another. A transition not defined in the state diagram 300 may be illegal. For example, a transition from S0 to S3 may be illegal because there is no applicable operation that can trigger such a transition. Likewise, applying operation F1 to state S0 may also be illegal because operation F1 is not applicable to state S0.

[0035] For illustrative purposes, the states and operations defined in the state diagram 300 may be denoted as:

$$s, s^* \in \left\{ \begin{array}{l} 0: \text{nil} \\ 1: \text{submitted} \\ 2: \text{issued} \\ 3: \text{pending modification} \\ 4: \text{end} \end{array} \right.$$

$$f \in \left\{ \begin{array}{l} 0: \text{submit} \\ 1: \text{approve} \\ 2: \text{modify} \\ 3: \text{reject} \end{array} \right.$$

Where s and s^* are the states before and after a transition, respectively, and f is the operation which triggers the transition from s to s^* .

[0036] As will be further described below, state diagrams such as the state diagram 300 may be utilized to verify validities of state transitions. If a user attempts to bypass certain operations or otherwise circumvent the transition rules defined in the state diagram 300, for example, the methods and devices disclosed herein can detect such attempts and prevent such attempts from being carried out on the blockchain 120. It is to be understood, however, that the references made to the state diagram 300 below are merely examples and are not meant to be limiting.

[0037] FIG. 4 illustrates a flow chart of a method 400 for protecting and verifying state information according to an embodiment. For illustrative purposes, a Blockchain, e.g., the blockchain 120 (FIG. 1), is depicted in FIG. 4. The Blockchain may be implemented to support various types of users, or parties, including, e.g., individuals, businesses, banks, financial institutions, hospitals, as well as other types of companies, organizations, and the like.

[0038] For illustrative purposes, a user, referred to as a Client, is depicted in FIG. 4. The Client may submit a record (e.g., an asset or a document) to the

Blockchain for recordation. The record may be of various formats and may contain various types of data. In some embodiments, the entire record submitted to the Blockchain may be encrypted. In some embodiments, the record submitted to the Blockchain may be partially encrypted. In some embodiments, the record submitted to the Blockchain may be in plaintext.

[0039] Continue with the example above, suppose that the Client is interested in issuing the record, and further suppose that the Blockchain enforces the state transition rules defined in the state diagram 300 to manage the state of the record during this issuing process, then each time the Client attempts to change the current state of the record, denoted as state s , the Blockchain may require the Client to submit information concerning an operation f to be performed and indicate what the next state of the record, denoted as state s^* , ought to be after the performance of the operation f . In this manner, if the Client submits information identifying an operation f not defined in the state diagram 300, or if the identified operation f is not applicable to the current state s of the record (e.g., if the Client identified F0 as the operation when the current state of the record is S1), the Blockchain may refuse to allow the Client to proceed. Similarly, if there is no valid transition from the current state s to the next state s^* (e.g., if the Client indicated S2 as the next state when the current state of the record is S0), or if applying the identified operation f to the current state s does not transition the current state s to the next state s^* (e.g., if the client identified F0 as the operation and indicated S2 as the next state when the current state of the record is S0), the Blockchain may refuse to allow the Client to proceed.

[0040] In some embodiments, to preserve privacy, the Blockchain may allow the Client to encrypt or otherwise protect the information concerning the current state s and the next state s^* , and submit only the protected state information to the

Blockchain. In this manner, the Blockchain can effectively reduce the likelihood of a malicious party being able to observe the state information associated with the record and gain insights into the record. However, allowing the Client to protect state information may make it relatively difficult for the Blockchain to perform the verification process and enforce state transition rules as described above. This lack of ability to enforce state transition rules may provide an opportunity for the owner of the record to manipulate the state of the record, e.g., by unilaterally changing the state from “submitted” to “issued,” thereby fraudulently bypassing the actual approval process depicted in FIG. 3. Furthermore, because the current state s and the next state s^* are both protected (e.g., encrypted), no one else may be able to detect the fraudulent bypass performed by the owner.

[0041] Therefore, in some embodiments, if the Client chooses to submit only the protected state information to the Blockchain, the Blockchain may require the Client to also submit a proof to the Blockchain to prove that the protected state information represents a valid state s^* that corresponds to what the next state of the record ought to be after the performance of a valid operation f . In some embodiments, the Blockchain may be willing to accept the proof without requiring the Client to disclose information concerning the current state s and the next state s^* in plaintext, provided that the Client agrees to generate the proof in accordance with the process depicted in FIG. 4.

[0042] At step 402, the Client may generate protected state information concerning the current state s and the next state s^* . In some embodiments, the Client may generate the protected state information using an encryption scheme. Alternatively, or additionally, the Client may generate the protected state information using a commitment scheme (e.g., Pedersen commitment) or a one-way function.

One of ordinary skill in the art will understand that a function $g: \mathcal{D} \rightarrow \mathcal{R}$ is one-way if, given a random element $x \in \mathcal{R}$, it is hard to compute a $y \in \mathcal{D}$ such that $g(y) = x$. In other words, it is difficult to compute a value of an input variable of a one-way function from a value of an output variable of the one-way function, making the function practically infeasible to invert and, thus, the function is called “one-way.” Hash functions, such as SHA 256 and the like, are examples of one-way functions.

[0043] For example, suppose that the Blockchain enforces state transition rules defined in a state diagram that defines a total of N states and M operations (e.g., the state diagram 300 defines $N = 5$ states and $M = 4$ operations), the Client may generate the protected state information concerning the current state s as a protected current state value $S = E(s, r_1), s \in [0, N - 1]$. The Client may also generate the protected state information concerning the next state s^* as a protected next state value $S^* = E(s^*, r_2), s^* \in [0, N - 1]$. It is to be understood that $E()$ may represent an encryption or commitment scheme, or a one-way function, chosen by the Client to protect information concerning the current state s and the next state s^* . r_1 and r_2 may represent random numbers that may be used in $E()$.

[0044] The Client may also generate a proof at step 402. The proof may be generated to prove to the Blockchain that the protected next state value S^* corresponds to a valid next state s^* that can be transitioned from the current state s after the performance of a valid operation .

[0045] In some embodiments, the generation of this proof may depend on the set of transition rules defined in the state diagram, e.g., the state diagram 300, enforced by the Blockchain. This set of transition rules, denoted as Ω , may be expressed in the format of $(s, f) \rightarrow s^*$, representing application of operation f to the current state of s to transition the current state s to the next state s^* . Thus, the set of

transition rules Ω defined in the state transition diagram 300 may be expressed as:

$$\Omega = \left\{ \begin{array}{l} (s_0, f_0) \rightarrow s_1, \\ (s_1, f_1) \rightarrow s_2, \\ (s_1, f_2) \rightarrow s_3, \\ (s_1, f_3) \rightarrow s_4, \\ (s_2, f_2) \rightarrow s_3, \\ (s_3, f_1) \rightarrow s_2, \\ (s_3, f_3) \rightarrow s_4 \end{array} \right\}$$

[0046] In some embodiments, the Client may generate the proof as a zero-knowledge proof, which refers to techniques that allow a prover to prove to a verifier that a statement is true without revealing any information beyond the validity of the statement itself. The Client may attempt to prove this by indicating that: (1) the Client knows the plaintext value of the current state s , and if r_1 is used to generate the protected current state value S , the Client knows the plaintext value of r_1 ; (2) the Client knows the plaintext value of the next state s^* , and if r_2 is used to generate the protected next state value S^* , the Client knows the plaintext value of r_2 ; and (3) the operation f is a valid operation that can transition the current state s to the next state s^* .

[0047] In some embodiments, to prove that the Client knows the plaintext values of s and r_1 , the Client may prove to the Blockchain that $S = E(s, r_1)$. To prove that the Client knows the plaintext values of s^* and r_2 , the Client may prove to the Blockchain that $S^* = E(s^*, r_2)$. Furthermore, to prove that the operation f is a valid operation that can transition the current state s to the next state s^* , the Client may prove to the Blockchain that $(s, f) \rightarrow s^*$ is a member of the set of transition rules Ω .

[0048] In some embodiments, the Client and the Blockchain may agree to implement zero-knowledge proof techniques such as Zero-Knowledge Succinct Non-

Interactive Argument of Knowledge (zk-SNARK) or the like. The Client may prove to the Blockchain that the Client knows a secret input w such that for a public input x , a certain relationship between x and w holds true. In some embodiments, the relationship may be defined as an arithmetic circuit. In some embodiments, the arithmetic circuit may be defined based on an equation of polynomials that can be evaluated based on x and w . In some embodiments, a proving key and a verification key may be generated in a setup phase based on the arithmetic circuit and one or more security parameters established for the zero-knowledge proof. One of ordinary skill in the art will understand that the setup phase may be executed by a trusted party, or by multiple independent parties collaboratively using a multi-party computation.

[0049] In some embodiments, the Client may set the public input x to a value generated based on the plaintext value of f and the protected values S and S^* , and set the secret input w to a value generated based on the plaintext values of s , s^* , r_1 , and r_2 . For example, the Client may set the value of x by concatenating f , S , and S^* together and set the value of w by concatenating s , s^* , r_1 , and r_2 together. In this manner, the Client may generate the proof using the secret and public inputs as well as the proving key to prove to the Blockchain that the Client is in possession of the secret input w . For example, if the relationship between the public input x and the secret input w can be defined based on an equation of polynomials, e.g., $a(x)w(x) = b(x)c(x)$, then $a()$ may be setup as the proving key and $b()c()$ may be setup as the verification key. The Client may generate the proof by calculating the polynomial $a(x)w(x)$ to prove to the Blockchain that the Client is in possession of the secret input w .

[0050] In some embodiments, the Blockchain may verify the proof using the public input x and the verification key. Continuing with the example above, the

Blockchain may verify the proof by determining whether the equation of polynomials $a(x)w(x) = b(x)c(x)$ holds true. For example, the Blockchain may compute the value of the polynomial $b(x)c(x)$ and check whether the value equals the proof submitted by the Client. If the equation holds true, then the Blockchain may accept the Client's proof that the Client is in possession of the secret input w , and the Blockchain may accept the Client's statements as true. However, if the equation does not hold true, the Blockchain may refuse to accept the Client's proof.

[0051] It is to be understood that the polynomial equation, the proving key, and the verification key described above are merely presented as examples and are not meant to be limiting. It is contemplated that the relationship between the public input x and the secret input w can be defined using various types of equations, and that the proving key and the verification key may be defined accordingly based on the relationship between the public input x and the secret input w . In some embodiments, the proof may also be translated into sigma protocols, which can be transformed into non-interactive zero-knowledge proofs of knowledge using Fiat-Shamir transform.

[0052] At step 404, the Client may submit a transaction to the Blockchain with payload containing $\{\text{record_ID}, S^*, f, \text{proof}\}$, where the record_ID is used to identify the record to be updated. For illustrative purposes, this transaction may be referred to as an "UPDATE" transaction, which may be used by the Client to update the state of the record identified by record_ID . It is contemplated that the Client may also include additional data fields in the UPDATE transaction. Such data fields may include, e.g., the content of the record or updates to the content of the record.

[0053] At step 406, the Blockchain may check the proof contained in the UPDATE transaction to determine whether the Client is in possession of the secret input w . In some embodiments, the Blockchain may utilize one or more smart

contracts executing on the Blockchain to provide the determination. Smart contracts are computer protocols implemented in the form of computer code that are incorporated into the Blockchain, to facilitate, verify, or enforce the negotiation or performance of contracts. For example, users of the Blockchain 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, e.g., to perform a transaction. Also for example, the smart contract may include a plurality of subroutines or functions, each of which may be a sequence of program instructions that performs a specific task. The smart contract may be operational code that is fully or partially executed without human interaction.

[0054] In some embodiments, a smart contract may be incorporated into the Blockchain to determine whether the proof is acceptable. The smart contract may verify the proof using the public input and the verification key. If the proof cannot be verified, the smart contract may refuse to allow the Client to proceed further. On the other hand, if the proof can be verified, then the smart contract may determine that the proof is acceptable and, at step 408, proceed to set the current state of the record recorded on the Blockchain to S^* as specified by the Client. Subsequent to step 408, s^* becomes the current state of the record and the protected value of s^* , denoted as S^* in the descriptions above, becomes the current state of the record as recorded on the Blockchain.

[0055] In some embodiments, the process depicted in FIG. 4 may be repeated each time the Client attempts to update the state of the record. In this manner, the Blockchain can detect invalid state transitions and prevent fraudulent attempts by the Client to bypass certain states even if the state information is encrypted or otherwise

protected by the Client. Also, in this manner, the Blockchain may allow the public to verify the validity of state transitions performed on a record, but the public may not be able to decipher what the record's current or previous states are, thereby preserving the Client's privacy.

[0056] FIG. 5 illustrates a flow chart of a method 500 for protecting and verifying state information of a record, according to an embodiment. The method 500 may be performed by one or more nodes in a blockchain system, e.g., the nodes 102-110 in the blockchain system 100 (FIG. 1). The nodes 102-110 in the blockchain system 100 may perform operations on a blockchain, e.g., the blockchain 120 (FIG. 1). The blockchain 120 may be implemented as the Blockchain in the examples described above.

[0057] At step 502, a node, e.g., the node 102, may receive a transaction submitted by a user. The user may be, e.g., the Client (FIG. 4), and the transaction submitted by the user may include an "UPDATE" transaction (FIG. 4). In some embodiments, the transaction may include an identification of a record (e.g., the `record_ID`), a protected state value (e.g., S^*) representing a state s^* of the record after performing an operation f on the record, and a proof prepared by the user. In some embodiments, the protected state value S^* may be generated (e.g., using an encryption scheme, a commitment scheme, a one-way function, or the like) based on a plaintext value representing the state s^* . In some embodiments, the node 102 may not receive the plaintext value of the state s^* from the user.

[0058] At step 504, the node 102 may determine whether the proof is acceptable. In some embodiments, the node 102 may determine whether the proof is acceptable based on whether the proof indicates that the user is in possession of information, such as the secret input w , which is generated at least partially based on

the plaintext value of the state s^* . In some embodiments, the proof may be a zero-knowledge proof. In such embodiments, the node 102 may determine whether the proof is acceptable without accessing the plaintext value of the state s^* . In some embodiments, the proof may be generated by the user to prove that the protected state value S^* represents a valid second (next) state s^* that can be transitioned from a valid first (current) state s after the performance of a valid operation f .

[0059] In some embodiments, the node 102 may determine whether the proof is acceptable based on whether the proof can show that the user has knowledge of the first state s of the record. The node 102 may also determine whether the proof is acceptable based on whether the proof can show that the user has knowledge of the second state s^* of the record. The node 102 may further determine whether the proof is acceptable based on whether the proof can show that the user has knowledge of the operation f to be performed with respect to the record. The node 102 may also determine whether the proof is acceptable based on whether the proof can show that the operation f is a valid operation that transitions the record from the first state s to the second state s^* .

[0060] In some embodiments, the user may prove to the node 102 that the user knows a secret input w such that for a public input x , a certain relationship between x and w holds true. The node 102 may determine whether the proof is acceptable by verifying the proof using the public input x and the verification key as described above. In some embodiments, if the node 102 accepts the user's proof that the user is in possession of the secret input w , the node 102 may accept the user's statements as true.

[0061] At step 506, in response to a determination that the proof is unacceptable, the node 102 may refuse to process the transaction submitted by the

user. In this manner, the node 102 can detect invalid state transitions and prevent fraudulent attempts by users to bypass certain states.

[0062] At step 508, in response to a determination that the proof is acceptable, the node 102 may set the recorded state of the record to the protected state value S^* . In this manner, the node 102 can verify the validity of state transitions performed on the record without the need to decipher the plaintext value of the operation f and the plaintext value of the state s^* . In some embodiments, the method 500 may be repeated each time the user attempts to update the state of the record.

[0063] FIG. 6 is a block diagram of a state transition verification apparatus 600, according to an embodiment. The apparatus 600 may be an implementation of a software process, and may correspond to the method 500 (FIG. 5). Referring to FIG. 6, the apparatus 600 may include a receiving module 602, a determination module 604, a rejection module 606, and a recording module 608.

[0064] The receiving module 602 may receive a transaction submitted by a user. The transaction may include an “UPDATE” transaction (FIG. 4), which may include an identification of a record (e.g., the record_ID), a protected state value (e.g., S^*) representing a state s^* of the record after performing an operation f on the record, and a proof prepared by the user. The receiving module 602 may provide the received transaction to the determining module 504.

[0065] The determining module 604 may determine whether the proof is acceptable. In response to a determination that the proof is unacceptable, the determining module 604 may request the rejection module 606 to refuse to process the transaction submitted by the user. Otherwise, the determining module 604 may provide the transaction to the recording module 608, which may set the recorded state of the record to the protected state value S^* . In this manner, the state transition

verification apparatus 600 can detect invalid state transitions and prevent fraudulent attempts by users to bypass certain states. The state transition verification apparatus 600 can also verify the validity of state transitions performed on the record without the need to decipher the plaintext value of the state s^* , thereby preserving the user's privacy.

[0066] 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 600 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.

[0067] For an implementation process of functions and roles of each module in the apparatus 600, references can be made to corresponding steps in the above-described methods. Details are omitted here for simplicity.

[0068] 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.

[0069] 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.

[0070] 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).

[0071] 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.

[0072] 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.

[0073] 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.

[0074] Although the specification has been described in conjunction with specific embodiments, many alternatives, modifications and variations will be apparent to those skilled in the art. Accordingly, the following claims embrace all such alternatives, modifications and variations that fall within the terms of the claims.

What Is Claimed Is:

1. A computer-implemented method for verifying state transition of a record, the method comprising:

receiving a transaction submitted by a user, the transaction comprising an identification of the record, a protected state value representing a state of the record after performing an operation on the record, and a proof prepared by the user;

determining whether the proof indicates that the user is in possession of information generated at least partially based on a plaintext value of the state of the record after performing the operation; and

in response to a determination that the proof indicates that the user is in possession of information, setting a recorded state of the record to the protected state value.

2. The method of claim 1, wherein the method is performed by one or more nodes in a blockchain system.

3. The method of claim 2, wherein the record comprises an asset or a document recorded on the blockchain system.

4. The method of claim 2, wherein the plaintext value of the state of the record after performing the operation is not received by the blockchain system.

5. The method of claim 4, wherein the determining whether the proof indicates that the user is in possession of information comprises determining whether the proof

indicates that the user is in possession of information without accessing the plaintext value of the state of the record after performing the operation.

6. The method of any preceding claim, wherein the proof is a zero-knowledge proof.

7. The method of any preceding claim, wherein the protected state value is generated based on the plaintext value representing the state of the record after performing the operation.

8. The method of claim 7, wherein the protected state value is generated by applying at least one of an encryption scheme, a commitment scheme, or a one-way function to the plaintext value representing the state of the record after performing the operation.

9. The method of any preceding claim, wherein the proof is prepared by the user to prove that the user has knowledge of a first state of the record, the first state of the record being a state of the record prior to performing the operation.

10. The method of claim 9, wherein the proof is prepared by the user to further prove that the user has knowledge of a second state of the record, the second state of the record being the state of the record after performing the operation.

11. The method of claim 10, wherein the proof is prepared by the user to further prove that the user has knowledge of the operation to be performed with respect to the record.

12. The method of claim 9, wherein the proof is prepared by the user to further prove that the operation is a valid operation that transitions the record from the first state to the second state.

13. A device for verifying state transition of a record, 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 12.

14. An apparatus for verifying state transition of a record, the apparatus comprising a plurality of modules for performing the method of any of claims 1 to 12.

15. 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 12.