

METHODS AND DEVICES FOR PREVENTING DENIAL-OF-SERVICE ATTACK ON BLOCKCHAIN SYSTEM

TECHNICAL FIELD

[0001] The specification relates generally to computer technologies, and more particularly, to methods and devices for preventing denial-of-service attacks on blockchain systems.

BACKGROUND

[0002] Blockchain systems, also known as distributed ledger systems (DLSs) or consensus systems, may enable participating entities 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] Due to the data structure that needs to be maintained, blockchain systems may have relatively low transaction rates. For example, public blockchain systems, such as Bitcoin and Ethereum, may only process around a dozen transactions per second. While consortium blockchain systems may support higher transaction rates, many are still limited to a few thousand transactions per second. Therefore, a relatively moderate increase in network traffic may cause data congestions, which may lead to breakdown of systems in certain circumstances. Consequently, blockchain systems may be prone to denial-of-service (DoS) attacks, including distributed denial-of-service (DDoS) attacks, which are malicious attempts to interrupt network traffic to target destinations, networks, or servers, by overwhelming the targets with massive amounts of fraudulent traffic from numerous source locations. Furthermore, blockchain systems often have multiple distributed nodes for receiving transactions, making blockchain systems more vulnerable to DDoS attacks.

SUMMARY

[0005] In one aspect, a computer-implemented method for preventing denial-of-service attacks on a blockchain system includes: retrieving a decryption key D_K from a k -th block of a blockchain; determining whether an encrypted transaction to be included in a block subsequent to the k -th block is valid based on the decryption key D_K ; in response to a determination that the encrypted transaction to be included in the block subsequent to the k -th block is invalid, rejecting the encrypted transaction; generating an encryption key E_{K+1} , generating a decryption key D_{K+1} based on the encryption key E_{K+1} ; generating a non-machine-readable representation of the encryption key E_{K+1} , the non-machine-readable representation

comprising a distorted image representing the encryption key E_{K+1} , and storing the decryption key D_{K+1} and the non-machine-readable representation of the encryption key E_{K+1} in the block subsequent to the k-th block.

[0006] In another aspect, a device for preventing denial-of-service attacks on a blockchain system 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: retrieve a decryption key D_K from a k-th block of a blockchain; determine whether an encrypted transaction to be included in a block subsequent to the k-th block is valid based on the decryption key D_K ; in response to a determination that the encrypted transaction to be included in the block subsequent to the k-th block is invalid, reject the encrypted transaction; generating an encryption key E_{K+1} , generating a decryption key D_{K+1} based on the encryption key E_{K+1} ; generating a non-machine-readable representation of the encryption key E_{K+1} , the non-machine-readable representation comprising a distorted image representing the encryption key E_{K+1} , and storing the decryption key D_{K+1} and the non-machine-readable representation of the encryption key E_{K+1} in the block subsequent to the k-th block.

[0007] In still another aspect, a non-transitory computer-readable medium have stored therein instructions that, when executed by a processor of a device, cause the device to perform a method for preventing denial-of-service attacks on a blockchain system. The method includes: retrieving a decryption key D_K from a k-th block of a blockchain; determining whether an encrypted transaction to be included in a block subsequent to the k-th block is valid based on the decryption key D_K ; in response to a determination that the encrypted transaction to be included in the block subsequent to the k-th block is invalid, rejecting the encrypted transaction;

generating an encryption key E_{K+1} , generating a decryption key D_{K+1} based on the encryption key E_{K+1} ; generating a non-machine-readable representation of the encryption key E_{K+1} , the non-machine-readable representation comprising a distorted image representing the encryption key E_{K+1} , and storing the decryption key D_{K+1} and the non-machine-readable representation of the encryption key E_{K+1} in the block subsequent to the k-th block.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] 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.

[0009] FIG. 1 is a schematic diagram of a blockchain system, according to an embodiment.

[0010] FIG. 2 is a schematic diagram of a computing device for implementing a node in a blockchain system, according to an embodiment.

[0011] FIG. 3 is a schematic diagram depicting a protocol for preventing denial-of-service attacks on a blockchain system, according an embodiment.

[0012] FIG. 4 is a flow chart of a method for preventing denial-of-service attacks on a blockchain system, according an embodiment.

[0013] FIG. 5 is a block diagram of an apparatus for preventing denial-of-service attacks on a blockchain system, according to an embodiment.

DETAILED DESCRIPTION

[0014] Embodiments of the specification provide methods and devices for preventing denial-of-service attacks on blockchain systems. The methods and devices may integrate an automated test into each block of a blockchain, thereby providing a mechanism for a blockchain system to determine whether a transaction is generated by a legitimate user or an automated script or program. The methods and devices may also implement the same test on all nodes across a blockchain, thereby providing fairness across all nodes.

[0015] Embodiments disclosed in the specification have one or more technical effects. In some embodiments, the methods and devices integrate an automated test into each block of a blockchain. This allows the methods and devices to determine, using the automated test, whether a transaction is generated by a legitimate user or an automated script or program. If it is determined that a transaction is generated by an automated script or program, that transaction may be discarded. In some embodiments, the methods and devices may implement the test by retrieving a decryption key from a k-th block of a blockchain and determining whether one or more encrypted transactions to be included in a block subsequent to the k-th block is valid based on the decryption key. This allows the methods and

devices to reject one or more encrypted transactions in response to a determination that the one or more encrypted transactions are invalid. In some embodiments, the methods and devices may implement the same test on all nodes across a blockchain. This allows the methods and devices to provide fairness across all nodes. In this manner, the methods and devices can be utilized in a blockchain system to ensure that every transaction submission has some human interactions, effectively protecting the blockchain system against denial-of-service attacks.

[0016] 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.

[0017] 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.

[0018] 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).

[0019] 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).

[0020] 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.

[0021] 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.

[0022] 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.

[0023] 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.

[0024] 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.

[0025] 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.

[0026] 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.

[0027] 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.

[0028] FIG. 3 illustrates a schematic diagram depicting a protocol for preventing denial-of-service attacks on a blockchain system, according to an embodiment. Referring to FIG. 3, each of Blocks K-1, K, and K+1 (K being an integer) may include a block head, which may include a timestamp and a cryptographic hash of a previous block, similar to the blocks in the blockchain 120 (FIG. 1). Blocks K-1, K, and K+1 may also include decryption keys D_{K-1} , D_K , and D_{K+1} , and representations of the encryption keys E_{K-1} , E_K , and E_{K+1} , respectively. In

some embodiments, the decryption keys and the representations of the encryption keys may be included as a part of their corresponding block heads.

[0029] In some embodiments, the decryption keys D_{K-1} , D_K , and D_{K+1} may be stored in a format that is readable to machines (e.g., a character string or a number), but the representations of the encryption keys E_{K-1} , E_K , and E_{K+1} may be stored in a format designed to be unreadable to machines. For instance, the representations of the encryption keys E_{K-1} , E_K , and E_{K+1} may be stored as images representing values of the encryption keys E_{K-1} , E_K , and E_{K+1} , but the images may be distorted in certain manners so that, e.g., only humans can read values of the encryption keys from the distorted images. In some embodiments, the distorted images may include CAPTCHA images, which are images commonly used in a type of challenge–response test that is designed to tell computers and humans apart. CAPTCHA is an acronym for "Completely Automated Public Turing test to tell Computers and Humans Apart."

[0030] In some embodiments, each of Blocks $K-1$, K , and $K+1$ may further include a data field, which may include one or more transactions denoted as $Tx1$ through Txn . In some embodiments, the transactions included in each block, e.g., Block $K+1$, may be encrypted using the encryption key provided in the previous block, e.g., E_K provided in Block K . The transactions encrypted in this manner may be referred to as encrypted transactions, which are denoted as $En(Tx1, E_K)$ through $En(Txn, E_K)$. It is to be understood that the value of "n" may vary amongst Blocks $K-1$, K , and $K+1$.

[0031] In some embodiments, for a user to submit a transaction to be included in a block, e.g., Block $K+1$, that user may be required to encrypt the transaction using the encryption key provided in the previous block, e.g., E_K provided in Block K .

However, because E_K is represented in a non-machine-readable format (e.g., a CAPTCHA image), only a human user may read the value of E_K and use the value of E_K to properly encrypt the transaction. A non-human user, e.g., an automated script, program, or a “bot,” may not be able to read the value of E_K , and therefore not be able to properly encrypt the transaction. In this manner, the mechanism for determining whether the transactions included in a block are generated by human users or automated scripts or programs may be executed automatically as the block is being formed or being added to the blockchain 120.

[0032] In some embodiments, transactions that are not properly encrypted may be discarded by the blockchain 120. For example, transactions that are not properly encrypted may be discarded by the nodes (e.g., nodes 102-110 depicted in FIG. 1) configured to operate on the blockchain 120. In some embodiments, all encrypted transactions included in Blocks K-1, K, and K+1 may be subject to a validation process. The validation process may be carried out when a node, e.g., the node 102, attempts to package one or more encrypted transactions to form a new block, e.g., Block K+1. The validation process may also be carried out when the node 102 participates in a consensus process to determine whether to accept a new block to be added to the blockchain 120.

[0033] For example, when a node, e.g., the node 102, attempts to package an encrypted transaction into a new block, e.g., Block K+1, the node 102 may invoke the validation process, which may in turn attempt to decrypt the encrypted transaction using the decryption key provided in the previous block, e.g., D_K provided in Block K. If the encrypted transaction can be decrypted successfully using D_K , then that encrypted transaction may be deemed valid. On the other hand, if the encrypted transaction cannot be decrypted successfully using D_K (e.g., if the encrypted

transaction cannot be decrypted using D_K at all or if the encrypted transaction is decrypted into something meaningless), then the validation process may conclude that the user who submitted the encrypted transaction did not use the correct encryption key E_K (e.g., because the user was not able to read the value of E_K properly or the user did not attempt to read the value of E_K at all). The validation process may therefore deem the transaction invalid, allowing the transaction to be discarded. In this manner, the node 102 may accept valid transactions and discard invalid transactions, effectively preventing invalid transactions from being packaged into Block $K+1$.

[0034] Similarly, when a node, e.g., the node 102, participates in a consensus process to determine whether to accept a new block, e.g., Block $K+1$, to be added to the blockchain 120, the node 102 may invoke the validation process, which may be utilized to decrypt all encrypted transactions included in Block $K+1$ using the decryption key provided in the previous block, e.g., D_K provided in Block K . In some embodiments, if the validation process deems one or more encrypted transactions included in Block $K+1$ to be invalid, Block $K+1$ may fail to reach a consensus and, therefore, not accepted as a new block to be added to the blockchain 120. On the other hand, if the validation process deems all encrypted transactions included in Block $K+1$ to be valid, the node 102 may accept Block $K+1$ as a new block to be added to the blockchain 120. In some embodiments, a threshold may be established, allowing Block $K+1$ to be accepted if it contains less than the threshold number of invalid transactions.

[0035] The encryption keys E_{K-1} , E_K , and E_{K+1} and the decryption keys D_{K-1} , D_K , and D_{K+1} may be generated using various types of key generation protocols. In some embodiments, the encryption keys may be generated based on a hash value

of a data structure (e.g., a Merkle tree) used to store the transactions recorded on the blockchain 120. The decryption keys may be generated based on the encryption keys. For example, suppose that a node, e.g., the node 102, is attempting to package a new block, e.g., Block K+1, to be added to the blockchain 120, the node 102 may generate the encryption key E_{K+1} for Block K+1 based on the root hash value (or the first m-bit of the root hash value) of the transaction Merkle tree currently recorded on the blockchain 120. In some embodiments, the node 102 may incorporate some randomness factors into the encryption key each time the encryption key is generated to further improve security. For example, the node 102 may generate the encryption key E_{K+1} for Block K+1 based on the root hash value plus a current time stamp, a random number, or the root hash of the to-be processed transactions (e.g., transactions that are being packaged by the node 102). Once the encryption key E_{K+1} is generated, the node 102 may generate the decryption key D_{K+1} and a non-machine-readable representation of the encryption key E_{K+1} accordingly based on the encryption key E_{K+1} .

[0036] In some embodiments, the node 102 may generate the decryption key D_{K+1} utilizing a key generation algorithm, including, e.g., the RSA (Rivest–Shamir–Adleman) algorithm and the like. The node 102 may generate the non-machine-readable representation of the encryption key E_{K+1} based on various techniques that can be utilized to tell computers and humans apart. For example, the node 102 may generate a distorted image, such as a CAPTCHA image, and utilize that distorted image as the non-machine-readable representation of the encryption key E_{K+1} .

[0037] In some embodiments, once the encryption key E_{K+1} and the decryption key D_{K+1} are generated, the node 102 may execute a decryption key validation process to determine whether the encryption key E_{K+1} and the decryption key D_{K+1}

form a valid pair (i.e., verify that the decryption key D_{K+1} can be used to properly decrypt a message encrypted using the encryption key E_{K+1}). In some embodiments, the node 102 may submit Block K+1 to the blockchain 120 after it has determined that the encryption key E_{K+1} and the decryption key D_{K+1} form a valid pair. Once the node 102 submits Block K+1 to the blockchain 120, other nodes may also execute the decryption key validation process as a part of the consensus process for deciding whether to accept Block K+1. In some embodiments, the node 102 may execute the decryption key validation process in a zero-knowledge proof manner, meaning that the node 102 may prove to the blockchain 120 that the encryption key E_{K+1} and the decryption key D_{K+1} form a valid pair using zero-knowledge proof. Zero-knowledge proof refers to techniques that allow a prover (e.g., the node 102) to prove to a verifier (e.g., the blockchain 120) that a statement is true without revealing any information beyond the validity of the statement itself. In this manner, the node 102 can prove that the encryption key E_{K+1} and the decryption key D_{K+1} form a valid pair without disclosing the encryption key E_{K+1} and the decryption key D_{K+1} themselves, further improving security. Once Block K+1 is accepted, the non-machine-readable representation of the encryption key E_{K+1} and the decryption key D_{K+1} may be recorded on the blockchain 120, allowing them to be utilized to encrypt and decrypt transactions in a subsequent block, e.g., Block K+2 (not shown), in the manner described above.

[0038] FIG. 4 illustrates a flow chart of a method 400 for preventing denial-of-service attacks on a blockchain system, according to an embodiment. The method 400 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).

[0039] At step 402, a node, e.g., the node 102, may retrieve a decryption key D_K from a k-th block of the blockchain 120. At step 404, the node 102 may determine whether an encrypted transaction to be included in a block subsequent to the k-th block is valid based on the decryption key D_K . In some embodiments, for a user to submit a transaction to be included in the block subsequent to the k-th block, that user may be required to encrypt the transaction using an encryption key E_K provided in the k-th block, which is a part of the blockchain 120 that is accessible to the user. In some embodiments, the encryption key E_K may be represented in a non-machine-readable format (e.g., a CAPTCHA image). The user may retrieve the CAPTCHA image from the blockchain 120 and obtain the value of the encryption key E_K based on the user's viewing of the CAPTCHA image. The user may then encrypt the transaction using the encryption key E_K . The encrypted transaction may then be validated at step 404.

[0040] At step 406, in response to a determination that the encrypted transaction to be included in the block subsequent to the k-th block is invalid, the node 102 may reject the encrypted transaction. For example, if the node 102 is attempting to package an encrypted transaction into the subsequent block, e.g., Block K+1, the node 102 may invoke the validation process previously described to decrypt the encrypted transaction using the decryption key D_K . If the encrypted transaction can be decrypted successfully using D_K , then that encrypted transaction may be deemed valid. On the other hand, if the encrypted transaction cannot be decrypted successfully using D_K (e.g., if the encrypted transaction cannot be decrypted using D_K at all or if the encrypted transaction is decrypted into something meaningless), then the validation process may conclude that the user who submitted

the encrypted transaction did not use the correct encryption key E_K (e.g., because the user was not able to read the value of E_K properly or the user did not attempt to read the value of E_K at all). The validation process may therefore deem the transaction invalid and reject the transaction, allowing the transaction to be discarded. In this manner, the node 102 may discard the encrypted transaction and prevent the encrypted transaction from being packaged into the subsequent block Block K+1.

[0041] Similarly, if the node 102 is participating in a consensus process to determine whether to accept a new block, e.g., Block K+1, to be added to the blockchain 120, the node 102 may invoke the validation process previously described to decrypt all encrypted transactions included in Block K+1 using the decryption key D_K . In some embodiments, if the validation process deems one or more encrypted transactions included in Block K+1 to be invalid, a consensus may fail to reach on Block K+1 and, therefore, the node 102 may refuse to accept Block K+1 as a new block to be added to the blockchain 120. In some embodiments, a threshold may be established, allowing the node 102 to accept Block K+1 if it contains less than the threshold number of encrypted transactions that are deemed invalid.

[0042] At step 408, the node 102 may generate an encryption key E_{K+1} for Block K+1. In some embodiments, the encryption key E_{K+1} may be generated based on a data structure utilized to store transactions recorded on the blockchain 120. In some embodiments, the encryption key E_{K+1} may be generated based on a hash value of the data structure.

[0043] At step 410, the node 102 may generate a decryption key D_{K+1} based on the encryption key E_{K+1} . At step 412, the node 102 may generate a non-machine-

readable representation of the encryption key E_{K+1} . In some embodiments, the non-machine-readable representation of the encryption key E_{K+1} may include a distorted image (e.g., a CAPTCHA image) representing the encryption key E_{K+1} . At step 414, the node 102 may store the decryption key D_{K+1} and the non-machine-readable representation of the encryption key E_{K+1} in Block K+1. In some embodiments, the decryption key D_{K+1} and the non-machine-readable representation of the encryption key E_{K+1} may be stored as a part of the block head of Block K+1, allowing them to be utilized to encrypt and decrypt transactions in a subsequent block, e.g., Block K+2, in the manner described above.

[0044] FIG. 5 is a block diagram of an apparatus 500 for preventing denial-of-service attacks on a blockchain system, according to an embodiment. The apparatus 500 may be an implementation of a software process, and may correspond to the method 400 (FIG. 4). Referring to FIG. 5, the apparatus 500 may include a retrieving module 502, a determining module 504, and a rejecting module 506.

[0045] The receiving module 502 may retrieve a decryption key D_K from a k-th block of the blockchain. The receiving module 502 may provide the decryption key D_K to the determining module 504.

[0046] The determining module 504 may determine whether an encrypted transaction to be included in a block subsequent to the k-th block, e.g., Block K+1, is valid based on the decryption key D_K . In response to a determination that the encrypted transaction to be included in Block K+1 is invalid, the determining module 504 may request the rejecting module 506 to reject the encrypted transaction.

[0047] In some embodiments, the apparatus 500 may also include a generating module 508 and a storing module 510.

[0048] The generating module 508 may generate an encryption key E_{K+1} for Block K+1. In some embodiments, the encryption key E_{K+1} may be generated based on a data structure utilized to store transactions recorded on the blockchain. In some embodiments, the encryption key E_{K+1} may be generated based on a hash value of the data structure. The generating module 508 may also generate a decryption key D_{K+1} based on the encryption key E_{K+1} and generate a non-machine-readable representation of the encryption key E_{K+1} . The generating module 508 may provide the decryption key D_{K+1} and the non-machine-readable representation of the encryption key E_{K+1} to the storing module 510, which may store the decryption key D_{K+1} and the non-machine-readable representation of the encryption key E_{K+1} in Block K+1.

[0049] It is to be understood that utilizing the methods and devices described above may allow the blockchain to impose the requirement of human interaction for every transaction submission, effectively protecting the blockchain against denial-of-service attacks generated by automated scripts, programs, or bots. Utilizing the methods and devices described above may also allow the blockchain to impose the same requirement on all nodes in the blockchain system, effectively providing fairness across all nodes in the blockchain system.

[0050] 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 500 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.

[0051] For an implementation process of functions and roles of each module in the apparatus 500, references can be made to corresponding steps in the above-described methods. Details are omitted here for simplicity.

[0052] 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.

[0053] 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.

[0054] 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).

[0055] 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.

[0056] 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.

[0057] 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.

[0058] 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 preventing denial-of-service attacks on a blockchain system, the method comprising:
 - retrieving a decryption key D_K from a k-th block of a blockchain;
 - determining whether an encrypted transaction to be included in a block subsequent to the k-th block is valid based on the decryption key D_K ;
 - in response to a determination that the encrypted transaction to be included in the block subsequent to the k-th block is invalid, rejecting the encrypted transaction;
 - generating an encryption key E_{K+1} ;
 - generating a decryption key D_{K+1} based on the encryption key E_{K+1} ;
 - generating a non-machine-readable representation of the encryption key E_{K+1} , the non-machine-readable representation comprising a distorted image representing the encryption key E_{K+1} ; and
 - storing the decryption key D_{K+1} and the non-machine-readable representation of the encryption key E_{K+1} in the block subsequent to the k-th block.
2. The method of claim 1, wherein the encryption key E_{K+1} is generated based on a data structure utilized to store transactions recorded on the blockchain.
3. The method of claim 2, wherein the encryption key E_{K+1} is generated based on a hash value of the data structure.
4. The method of claim 2 or 3, wherein rejecting the encrypted transaction further comprises:

preventing the encrypted transaction from being packaged into the block subsequent to the k-th block.

5. The method of any one of claims 2 to 4, wherein rejecting the encrypted transaction further comprises:

refusing to accept the block subsequent to the k-th block as a new block of the blockchain.

6. The method of claim 5, wherein refusing to accept the block subsequent to the k-th block as a new block of the blockchain further comprises:

determining a number of invalid encrypted transactions contained in the block subsequent to the k-th block; and

refusing to accept the block subsequent to the k-th block as a new block of the blockchain when the number of invalid encrypted transactions exceeds a threshold.

7. A device for preventing denial-of-service attacks on a blockchain system, 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 6.

8. An apparatus for preventing denial-of-service attacks on a blockchain system, the apparatus comprising a plurality of modules for performing the method of any of claims 1 to 6.

9. 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 6.