BLOCKCHAIN SMART CONTRACT-BASED TRANSACTION HASH ACQUISITION METHODS AND SYSTEMS

TECHNICAL FIELD

[0001] Embodiments of the present specification relate to the field of information technologies, and in particular, to transaction hash acquisition blockchain smart contract-based methods and systems.

BACKGROUND

[0002] Currently, deploying smart contracts in a blockchain network established based on the Ethereum protocol can satisfy various online service demands.

[0003] In practice, for some service demands, when each node in the blockchain network invokes, by using an Ethereum virtual machine, a smart contract corresponding to the service demands to execute a transaction, the node needs to obtain a transaction hash of the current transaction.

[0004] However, the existing Ethereum virtual machine does not support an operation of obtaining the transaction hash of the current transaction.

SUMMARY

[0005] To solve a problem that an existing Ethereum virtual machine does not support a transaction hash acquisition operation by default, embodiments of the present specification provide blockchain smart contract-based transaction hash acquisition methods and systems. Technical solutions are as follows:

[0006] According to a first aspect of the embodiments of the present specification, a blockchain smart contract-based transaction hash acquisition method is provided, where an instruction set of a blockchain virtual machine includes a transaction hash acquisition instruction, and transaction hash acquisition logic corresponding to the transaction hash acquisition instruction is deployed in the blockchain virtual machine; an instruction set of a smart contract compiler includes the transaction hash acquisition instruction, and a service smart contract compiled by using the smart contract compiler

includes the transaction hash acquisition instruction; the service smart contract is deployed in a blockchain network; and the transaction hash acquisition method includes the following: a node in the blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes; when executing the service initiation transaction, each node in the blockchain network invokes the service smart contract by using the blockchain virtual machine; and the node triggers execution of the transaction hash acquisition logic based on the transaction hash acquisition instruction in the service smart contract by using the blockchain virtual machine, to obtain a transaction hash of the service initiation transaction.

[0007] According to a second aspect of the embodiments of the present specification, another blockchain smart contract-based transaction hash acquisition method is provided, where an instruction set of a blockchain virtual machine includes a transaction hash acquisition instruction, and transaction hash acquisition logic corresponding to the transaction hash acquisition instruction is deployed in the blockchain virtual machine; an instruction set of a smart contract compiler includes the transaction hash acquisition instruction, a service smart contract compiled by using the smart contract compiler includes a contract identifier of a transaction hash acquisition smart contract, and the transaction hash acquisition smart contract is a smart contract pre-deployed in a blockchain network; the service smart contract is deployed in the blockchain network; and the transaction hash acquisition method includes the following: a node in the blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes; when executing the service initiation transaction, each node in the blockchain network invokes the service smart contract by using the blockchain virtual machine; the node invokes the transaction hash acquisition smart contract based on the contract identifier of the transaction hash acquisition smart contract in the service smart contract by using the blockchain virtual machine; and the node triggers execution of the transaction hash acquisition logic based on the transaction hash acquisition instruction in the transaction hash acquisition smart contract by using the blockchain virtual machine, to obtain a transaction hash of the service initiation transaction.

[0008] According to a third aspect of the embodiments of the present specification, a blockchain virtual machine is provided, configured to implement the method in the first aspect and the method in the second aspect, where an instruction set of the smart contract compiler includes a transaction hash acquisition instruction, and a service

smart contract compiled by using the smart contract compiler includes the transaction hash acquisition instruction.

[0009] According to a fourth aspect of the embodiments of the present specification, a smart contract compiler is provided, configured to implement the method in the first aspect, where an instruction set of the smart contract compiler includes a transaction hash acquisition instruction, and a service smart contract compiled by using the smart contract compiler includes the transaction hash acquisition instruction.

[0010] According to a fifth aspect of the embodiments of the present specification, a smart contract compiler is provided, configured to implement the method in the second aspect, where an instruction set of the smart contract compiler includes a transaction hash acquisition instruction, a service smart contract compiled by using the smart contract compiler includes a contract identifier of a transaction hash acquisition smart contract, and the transaction hash acquisition smart contract is a smart contract pre-deployed in a blockchain network.

[0011] According to a sixth aspect of the embodiments of the present specification, a blockchain system is provided, including a blockchain network, where an instruction set of a blockchain virtual machine includes a transaction hash acquisition instruction, transaction hash acquisition logic corresponding to the transaction hash acquisition instruction is deployed in the blockchain virtual machine, an instruction set of a smart contract compiler includes the transaction hash acquisition instruction, a service smart contract compiled by using the smart contract compiler includes the transaction hash acquisition instruction, and the service smart contract is deployed in the blockchain network; a node in the blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes; and when executing the service initiation transaction, each node in the blockchain network invokes the service smart contract by using the blockchain virtual machine, and triggers execution of the transaction hash acquisition logic based on the transaction hash acquisition instruction in the service smart contract by using the blockchain virtual machine, to obtain a transaction hash of the service initiation transaction.

[0012] According to a seventh aspect of the embodiments of the present specification, another blockchain system is provided, including a blockchain network, where an instruction set of a blockchain virtual machine includes a transaction hash acquisition instruction, transaction hash acquisition logic corresponding to the transaction hash acquisition instruction is deployed in the blockchain virtual machine,

an instruction set of a smart contract compiler includes the transaction hash acquisition instruction, a service smart contract compiled by using the smart contract compiler includes a contract identifier of a transaction hash acquisition smart contract, the transaction hash acquisition smart contract is a smart contract pre-deployed in the blockchain network, and the service smart contract is deployed in the blockchain network; a node in the blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes; and when executing the service initiation transaction, each node in the blockchain network invokes the service smart contract by using the blockchain virtual machine, invokes the transaction hash acquisition smart contract based on the contract identifier of the transaction hash acquisition smart contract in the service smart contract by using the blockchain virtual machine, and triggers execution of the transaction hash acquisition logic based on the transaction hash acquisition instruction in the transaction hash acquisition smart contract by using the blockchain virtual machine, to obtain a transaction hash of the service initiation transaction.

[0013] In the technical solutions provided in the embodiments of the present specification, in one aspect, the transaction hash acquisition instruction is defined and added to the instruction set of the blockchain virtual machine. In addition, the transaction hash acquisition logic corresponding to the transaction hash acquisition instruction is deployed in the blockchain virtual machine. In another aspect, the defined transaction hash acquisition instruction further needs to be added to the instruction set of the smart contract compiler, so the service smart contract compiled by using the smart contract compiler includes the transaction hash acquisition instruction.

[0014] As such, if the service smart contract is deployed in the blockchain network, when constructing the service initiation transaction, a user can specify to invoke the service smart contract to execute the service initiation transaction. Therefore, when executing the service initiation transaction, the virtual machine invokes the service smart contract, and triggers execution of the pre-deployed transaction hash acquisition logic based on the transaction hash acquisition instruction in the service smart contract.

[0015] In the embodiments of the present specification, the instruction sets of the blockchain virtual machine and the smart contract compiler are expanded, so the blockchain virtual machine can support the transaction hash acquisition operation by default.

[0016] It should be understood that the aforementioned general description and the

following detailed description are merely examples and illustrative, and cannot limit the embodiments of the present specification.

[0017] In addition, any one of the embodiments of the present specification does not need to achieve all the aforementioned effects.

BRIEF DESCRIPTION OF DRAWINGS

[0018] To describe the technical solutions in the embodiments of the present specification or in the existing technology more clearly, the following briefly describes the accompanying drawings needed for describing the embodiments or the existing technology. Clearly, the accompanying drawings in the following description merely show some embodiments of the present specification, and a person of ordinary skill in the art can still derive other drawings from these accompanying drawings.

[0019] FIG. 1 is a schematic flowchart illustrating a blockchain smart contract-based encoding method, according to some embodiments of the present specification;

[0020] FIG. 2 is a schematic flowchart illustrating another blockchain smart contract-based encoding method, according to some embodiments of the present specification;

[0021] FIG. 3 is a schematic flowchart illustrating a blockchain smart contract-based decoding method, according to some embodiments of the present specification;

[0022] FIG. 4 is a schematic flowchart illustrating another blockchain smart contract-based decoding method, according to some embodiments of the present specification;

[0023] FIG. 5a and FIG. 5b are schematic diagrams illustrating deployment of a BASE64 encoding/decoding operation, according to some embodiments of the present specification;

[0024] FIG. 6 is a schematic flowchart illustrating a blockchain smart contract-based signature verification method, according to some embodiments of the present specification;

[0025] FIG. 7 is a schematic flowchart illustrating another blockchain smart contract-based signature verification method, according to some embodiments of the present specification;

[0026] FIG. 8a and FIG. 8b are schematic diagrams illustrating deployment of an RSA signature verification operation, according to some embodiments of the present

specification;

[0027] FIG. 9 is a schematic flowchart illustrating a blockchain smart contract-based data processing method, according to some embodiments of the present specification;

[0028] FIG. 10 is a schematic flowchart illustrating a blockchain smart contract-based data processing method, according to some embodiments of the present specification;

[0029] FIG. 11a and FIG. 11b are schematic diagrams illustrating deployment of a JavaScript Object Notation (JSON) processing operation, according to some embodiments of the present specification;

[0030] FIG. 12 is a schematic flowchart illustrating a blockchain smart contract-based data processing method, according to some embodiments of the present specification;

[0031] FIG. 13 is a schematic flowchart illustrating a blockchain smart contract-based data processing method, according to some embodiments of the present specification;

[0032] FIG. 14a and FIG. 14b are schematic diagrams illustrating deployment of a processing operation, according to some embodiments of the present specification;

[0033] FIG. 15 is a schematic flowchart illustrating a blockchain smart contract-based transaction hash acquisition method, according to some embodiments of the present specification;

[0034] FIG. 16 is a schematic flowchart illustrating a blockchain smart contract-based transaction hash acquisition method, according to some embodiments of the present specification;

[0035] FIG. 17a and FIG. 17b are schematic diagrams illustrating deployment of a transaction hash acquisition operation, according to some embodiments of the present specification;

[0036] FIG. 18 is a schematic flowchart illustrating a blockchain smart contract-based transfer method, according to some embodiments of the present specification;

[0037] FIG. 19 is a schematic flowchart illustrating a blockchain smart contract-based transaction hash acquisition method, according to some embodiments of the present specification;

[0038] FIG. 20a and FIG. 20b are schematic diagrams illustrating deployment of a verification operation related to a transfer transaction, according to some embodiments of the present specification;

[0039] FIG. 21 is a schematic structural diagram illustrating a blockchain system, according to some embodiments of the present specification; and

[0040] FIG. 22 is a schematic structural diagram illustrating a computer device used to configure a method in some embodiments of the present specification.

DESCRIPTION OF EMBODIMENTS

[0041] In the present specification, a blockchain virtual machine is an execution program on which each blockchain node relies when executing a transaction. The blockchain virtual machine provides an execution environment for a blockchain transaction.

[0042] It is worthwhile to note that the blockchain virtual machine can be an Ethereum virtual machine (EVM) recorded in the Ethereum protocol, or can be a virtual machine in blockchain protocols other than the Ethereum protocol.

[0043] In the present specification, a smart contract compiler is a program used to compile a smart contract written in a programming language (such as the Solidity language) into a machine language (such as bytecode or binary code) that the blockchain virtual machine can recognize and execute. It is worthwhile to note that a smart contract deployed in a blockchain network is generally a smart contract compiled by using the smart contract compiler, namely, a smart contract in the form of bytecode or binary code.

[0044] In the Ethereum protocol or other blockchain protocols similar to the Ethereum protocol, the blockchain virtual machine locally maintains a mapping relationship (as shown in Table 1 below) between several groups of instructions and operations, and code logic of each operation in Table 1 is further locally deployed in the blockchain virtual machine. Instruction 1 to instruction N are instructions supported by the blockchain virtual machine by default, and instruction 1 to instruction N form an instruction set of the blockchain virtual machine. When invoking a smart contract, if reading any one of the instructions in Table 1 from the smart contract, the blockchain virtual machine triggers execution of code logic of an operation corresponding to the instruction. It is worthwhile to note that the instruction described in the present specification is generally in the form of bytecode or binary code.

Instruction 1	Operation 1

Instruction 2	Operation 2
Instruction 3	Operation 3
...	...
Instruction N	Operation N

Table 1

[0045] Therefore, if the blockchain virtual machine is expected to perform any one of the operations in Table 1, an instruction corresponding to the operation needs to be written in advance into a smart contract to be invoked by the blockchain virtual machine. It means that when compiling the smart contract in advance, the smart contract compiler can compile, into a corresponding instruction, content that is declared in the smart contract (written in the programming language) to implement any one of the operations in Table 1, so the blockchain virtual machine can recognize the instruction, which requires the smart contract compiler also to support instruction 1 to instruction N in Table 1 by default, and instruction 1 to instruction N form an instruction set of the smart contract compiler.

[0046] In summary, in the Ethereum protocol or other blockchain protocols similar to the Ethereum protocol, the blockchain virtual machine, the smart contract compiler, and the smart contract generally cooperate to implement a specific service demand.

[0047] However, an existing EVM locally supports limited operations by default, but service demands are complex and diverse. For example, the existing EVM and Ethereum smart contract compiler do not support the following operations by default, and consequently, some service demands that depend on the following operations cannot be implemented in an existing Ethereum architecture:

[0048] 1. Encoding/decoding operation based on 64 printable characters (BASE64).

[0049] 2. RSA signature verification operation: In 1977, the RSA algorithm was proposed by Ronald Ron Rivest, Adi Shamir, and Leonard Adleman, and the RSA algorithm was thus named.

[0050] 3. Processing operation of JavaScript Object Notation (JSON) data: JSON is a common data exchange format, and JSON data is data having the data exchange format.

[0051] 4. Processing operation of extensible markup language (XML) data: XML is a common data exchange format, and XML data is data having the data exchange format.

[0052] 5. Operation of obtaining a transaction hash of a currently executed

transaction.

[0053] 6. For a transfer transaction, when asset balance of a transferor and a transfer amount are not exposed, it is determined whether the asset balance of the transferor is sufficient to cover the transfer amount of the transfer transaction.

[0054] The existing EVM and Ethereum smart contract compiler do not support the aforementioned operations by default. Therefore, in the existing technologies, code logic for implementing any one of the aforementioned operations is generally written into the smart contract directly, so when executing the transaction, the EVM invokes the smart contract to execute the code logic of the operation. That is, although the EVM does not support the aforementioned operations by default, the EVM can also perform the aforementioned operations by writing the logic code of the aforementioned operations into the smart contract and enabling the EVM to invoke the smart contract.

[0055] However, in practice, the EVM is less efficient in executing the code logic in the smart contract than directly executing the locally pre-deployed code logic.

[0056] A core idea of the present application is as follows: In one aspect, the instruction set of the blockchain virtual machine is expanded to add instructions corresponding to the aforementioned operations, and the code logic corresponding to the aforementioned operations are locally pre-deployed in the blockchain virtual machine, so the blockchain virtual machine can support the aforementioned operations by default. In another aspect, the instruction set of the smart contract compiler is also expanded to add the instructions corresponding to the aforementioned operations. For the same operation, an instruction that corresponds to the operation and that is added to the instruction set of the blockchain virtual machine should be consistent with an instruction that corresponds to the operation and that is added to the instruction set of the smart contract compiler. Details are shown in Table 2.

Instruction 1	Operation 1
Instruction 2	Operation 2
Instruction 3	Operation 3
BASE64 encoding instruction	BASE64 encoding operation
...	...
Instruction N	Operation N

Table 2

[0057] As such, in an example of the aforementioned BASE64 encoding operation,

when the smart contract is written in the programming language, invoking of the BASE64 encoding operation is declared in the smart contract, so the smart contract compiler compiles the declaration into the BASE64 encoding instruction when compiling the smart contract. After the smart contract is deployed in the blockchain network, a user can specify to invoke the smart contract in a service initiation transaction when initiating a service. As such, when executing the service initiation transaction, the blockchain virtual machine invokes the smart contract, and triggers execution of corresponding BASE64 encoding logic locally when reading the BASE64 encoding instruction from the smart contract, to implement the BASE64 encoding operation.

[0058] To make a person skilled in the art better understand the technical solutions in the embodiments of the present specification, the following describes in detail the technical solutions in the embodiments of the present specification with reference to the accompanying drawings in the embodiments of the present specification. Clearly, the described embodiments are merely some but not all of the embodiments of the present specification. All other embodiments obtained by a person of ordinary skill in the art based on the embodiments of the present specification shall fall within the protection scope of the present specification.

[0059] The technical solutions provided in the embodiments of the present specification are described in detail below with reference to the accompanying drawings. It is worthwhile to note that because the following embodiments are based on similar technical ideas, the following embodiments can be understood with reference to each other.

[0060] Embodiment 1

[0061] FIG. 1 is a schematic flowchart illustrating a blockchain smart contract-based encoding method, according to some embodiments of the present specification. The method includes the following steps:

[0062] S100: A node in a blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0063] S102: When executing the service initiation transaction, each node in the blockchain network invokes a service smart contract by using a blockchain virtual machine.

[0064] S104: The node triggers execution of BASE64 encoding logic based on a BASE64 encoding instruction in the service smart contract by using the blockchain

virtual machine, to perform an encoding operation on data to be encoded.

[0065] In some embodiments, the blockchain network needs to be pre-configured, so the following can be implemented:

[0066] (1) An instruction set of the blockchain virtual machine includes the BASE64 encoding instruction, and the BASE64 encoding logic corresponding to the BASE64 encoding instruction is deployed in the blockchain virtual machine.

[0067] (2) An instruction set of a smart contract compiler includes the BASE64 encoding instruction, and the service smart contract compiled by using the smart contract compiler includes the BASE64 encoding instruction.

[0068] (3) The service smart contract is deployed in the blockchain network.

[0069] The service smart contract is a smart contract corresponding to a service that needs to invoke a BASE64 encoding function.

[0070] In the embodiments of the present specification, the blockchain network includes a plurality of nodes. From the perspective of software, a node is a blockchain program used to implement a blockchain function. From the perspective of hardware, a node is user device installing with a blockchain program. In practice, each node can be connected to at least one client device (or wallet), and a transaction in the blockchain is usually constructed by the client device.

[0071] The transaction described in the embodiments of the present specification is a piece of data that is created by a user by using the client device in the blockchain and that needs to be finally published to a distributed database in the blockchain. The transaction is a data structure agreed in the blockchain protocol. A piece of data needs to be encapsulated into a transaction before the data is stored in the blockchain.

[0072] Transactions in the blockchain include a transaction in a narrow sense and a transaction in a broad sense. The transaction in the narrow sense is value transfer published by the user to the blockchain. For example, in a conventional Bitcoin blockchain network, a transaction can be transfer initiated by the user in the blockchain. The transaction in the broad sense is service data that is published by the user to the blockchain and that has a service intention. For example, an operator can establish a consortium blockchain based on actual service demands, and deploy some other types of online services (such as a house rental service, a vehicle scheduling service, an insurance claim service, a credit service, and a medical service) that are not related to value transfer in the consortium blockchain. In the consortium blockchain, a transaction can be a service message or a service request that is published by the user in the

consortium blockchain and that has a service intention.

[0073] In the blockchain network, the user usually initiates a service in the form of a transaction. Specifically, the node needs to obtain the service initiation transaction if the aforementioned service smart contract is not the only smart contract deployed in the blockchain network, and the service initiation transaction further indicates a contract identifier of the service smart contract, to specify the smart contract that needs to be invoked when the transaction is executed.

[0074] The service initiation transaction is usually constructed and sent by the user to the node by using the client device. Generally, client devices connected in the blockchain network are in a one-to-one correspondence with users.

[0075] After the node broadcasts the service initiation transaction to other nodes, each node receives the service initiation transaction, and then needs to invoke, by using the blockchain virtual machine, the service smart contract to execute the service initiation transaction. It is worthwhile to note here that in the blockchain network, a blockchain virtual machine is deployed on each node. When the node executes a transaction, the blockchain virtual machine deployed on the node actually executes the transaction.

[0076] In Embodiment 1, after invoking the service smart contract, the blockchain virtual machine reads bytecode or binary code in the service smart contract, and executes the bytecode or the binary code. When the blockchain virtual machine reads the BASE64 encoding instruction in the service smart contract, it is equivalent to determining that a BASE64 encoding operation needs to be performed. Therefore, the blockchain virtual machine triggers execution of the locally pre-deployed BASE64 encoding logic, to perform an encoding operation on the data to be encoded.

[0077] In practice, depending on a specific service demand, the data to be encoded can be included in the service initiation transaction, can be included in the service smart contract, or can be generated when the blockchain virtual machine executes the service initiation transaction.

[0078] Furthermore, in addition to executing the service initiation transaction by using the blockchain virtual machine, each node further needs to write the service initiation transaction into the blockchain based on a consensus algorithm.

[0079] In addition, it is worthwhile to that if the blockchain virtual machine triggers execution of the BASE64 encoding logic based on the BASE64 encoding instruction in the service smart contract, a stack-based parameter passing method (that is, data

obtained after BASE64 encoding is usually written into a stack) is usually used in an execution process for data transfer. However, a data length of the data obtained after BASE64 encoding is not fixed. If the stack-based parameter passing method is used, a corresponding BASE64 encoding instruction needs to be predetermined for each possible data length. This is relatively complex to implement.

[0080] Therefore, the following Embodiment 2 provides another blockchain smart contract-based encoding method.

[0081] Embodiment 2

[0082] FIG. 2 is a schematic flowchart illustrating another blockchain smart contract-based encoding method, according to some embodiments of the present specification. The method includes the following steps:

[0083] S200: A node in a blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0084] S202: When executing the service initiation transaction, each node in the blockchain network invokes a service smart contract by using a blockchain virtual machine.

[0085] S204: The node invokes a BASE64 encoding smart contract based on a contract identifier of the BASE64 encoding smart contract in the service smart contract by using the blockchain virtual machine.

[0086] S206: The node triggers execution of BASE64 encoding logic based on a BASE64 encoding instruction in the BASE64 encoding smart contract by using the blockchain virtual machine, to perform an encoding operation on data to be encoded.

[0087] In some embodiments, the blockchain network needs to be pre-configured, so the following can be implemented:

[0088] (1) An instruction set of the blockchain virtual machine includes the BASE64 encoding instruction, and the BASE64 encoding logic corresponding to the BASE64 encoding instruction is deployed in the blockchain virtual machine.

[0089] (2) An instruction set of a smart contract compiler includes the BASE64 encoding instruction, the service smart contract compiled by using the smart contract compiler includes the contract identifier of the BASE64 encoding smart contract, and the BASE64 encoding smart contract is a smart contract pre-deployed in the blockchain network.

[0090] (3) The service smart contract is deployed in the blockchain network.

[0091] A difference between Embodiment 2 and Embodiment 1 mainly lies in the

following: In Embodiment 2, after the blockchain virtual machine invokes the service smart contract, when the blockchain virtual machine reads the contract identifier of the BASE64 encoding smart contract, it is equivalent to determining that the BASE64 encoding smart contract needs to be further invoked. The blockchain virtual machine invokes the BASE64 encoding smart contract, and also reads bytecode or binary code in the BASE64 encoding smart contract. When the blockchain virtual machine reads the BASE64 encoding instruction, it is equivalent to determining that a BASE64 encoding operation needs to be performed. Therefore, the blockchain virtual machine triggers execution of the locally pre-deployed BASE64 encoding logic, to perform an encoding operation on the data to be encoded.

[0092] That is, in Embodiment 2, when the smart contract compiler compiles the service smart contract, if the smart contract compiler identifies that invoking of the BASE64 encoding operation is declared in the service smart contract, the smart contract compiler compiles the declaration into the contract identifier of the BASE64 encoding smart contract instead of the BASE64 encoding instruction. As such, when invoking the service smart contract, the blockchain virtual machine further invokes the BASE64 encoding smart contract.

[0093] In the field of blockchain technologies, the BASE64 encoding smart contract is actually a precompiled contract. When invoking and executing a precompiled contract such as the BASE64 encoding smart contract, the blockchain virtual machine does not transfer a parameter by using a stack-based parameter passing method, but transfers the parameter by using a memory-based parameter passing method (which supports reading and writing of data having an unfixed length).

[0094] Embodiment 3

[0095] FIG. 3 is a schematic flowchart illustrating a blockchain smart contract-based encoding method, according to some embodiments of the present specification. The method includes the following steps:

[0096] S300: A node in a blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0097] S302: When executing the service initiation transaction, each node in the blockchain network invokes a service smart contract by using a blockchain virtual machine.

[0098] S304: The node triggers execution of BASE64 decoding logic based on a BASE64 decoding instruction in the service smart contract by using the blockchain

virtual machine, to perform a decoding operation on data to be decoded.

[0099] In some embodiments, the blockchain network needs to be pre-configured, so the following can be implemented:

[0100] (1) An instruction set of the blockchain virtual machine includes the BASE64 decoding instruction, and the BASE64 decoding logic corresponding to the BASE64 decoding instruction is deployed in the blockchain virtual machine.

[0101] (2) An instruction set of a smart contract compiler includes the BASE64 decoding instruction, and the service smart contract compiled by using the smart contract compiler includes the BASE64 decoding instruction.

[0102] (3) The service smart contract is deployed in the blockchain network.

[0103] Because the BASE64 decoding operation and the BASE64 encoding operation are a group of operations that correspond to each other, for related descriptions, references can be made to Embodiment 1. Details are not omitted for simplicity.

[0104] It is worthwhile to note that in practice, depending on a specific service demand, the data to be decoded can be included in the service initiation transaction, can be included in the service smart contract, or can be generated when the blockchain virtual machine executes the service initiation transaction.

[0105] In addition, the aforementioned problem that implementation of stack-based parameter passing is complex also exists in Embodiment 3. Therefore, the following Embodiment 4 provides another blockchain smart contract-based decoding method.

[0106] Embodiment 4

[0107] FIG. 4 is a schematic flowchart illustrating another blockchain smart contract-based decoding method, according to some embodiments of the present specification. The method includes the following steps:

[0108] S400: A node in a blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0109] S402: When executing the service initiation transaction, each node in the blockchain network invokes a service smart contract by using a blockchain virtual machine.

[0110] S404: The node invokes a BASE64 decoding smart contract based on a contract identifier of the BASE64 decoding smart contract in the service smart contract by using the blockchain virtual machine.

[0111] S406: The node triggers execution of BASE64 decoding logic based on a

BASE64 decoding instruction in the BASE64 decoding smart contract by using the blockchain virtual machine, to perform a decoding operation on data to be decoded.

[0112] In some embodiments, the blockchain network needs to be pre-configured, so the following can be implemented:

[0113] (1) An instruction set of the blockchain virtual machine includes the BASE64 decoding instruction, and the BASE64 decoding logic corresponding to the BASE64 decoding instruction is deployed in the blockchain virtual machine.

[0114] (2) An instruction set of a smart contract compiler includes the BASE64 decoding instruction, the service smart contract compiled by using the smart contract compiler includes the contract identifier of the BASE64 decoding smart contract, and the BASE64 decoding smart contract is a smart contract pre-deployed in the blockchain network.

[0115] (3) The service smart contract is deployed in the blockchain network.

[0116] A difference between Embodiment 4 and Embodiment 3 mainly lies in the following: In Embodiment 4, after the blockchain virtual machine invokes the service smart contract, when the blockchain virtual machine reads the contract identifier of the BASE64 decoding smart contract, it is equivalent to determining that the BASE64 decoding smart contract needs to be further invoked. The blockchain virtual machine invokes the BASE64 decoding smart contract, and also reads bytecode or binary code in the BASE64 decoding smart contract. When the blockchain virtual machine reads the BASE64 decoding instruction, it is equivalent to determining that a BASE64 decoding operation needs to be performed. Therefore, the blockchain virtual machine triggers execution of the locally pre-deployed BASE64 decoding logic, to perform a decoding operation on the data to be decoded.

[0117] FIG. 5a is a schematic diagram illustrating deployment of a BASE64 encoding/decoding operation corresponding to Embodiment 1 and Embodiment 3, according to some embodiments of the present specification. As shown in FIG. 5a, first, at least one of the BASE64 encoding instruction or the BASE64 decoding instruction is added to the instruction set of the blockchain virtual machine, and at least one of the BASE64 encoding logic or the BASE64 decoding logic is deployed in the blockchain virtual machine. Second, at least one of the BASE64 encoding instruction or the BASE64 decoding instruction is added to the instruction set of the smart contract compiler. Third, the service smart contract (which includes at least one of the BASE64 encoding instruction or the BASE64 decoding instruction) compiled by using the smart

contract compiler is deployed in the blockchain network.

[0118] It is worthwhile to note that when a function instruction (which includes the various instructions described in the present specification) is added to the smart contract compiler, processing logic (such as parameter/returning value processing logic) corresponding to the function instruction generally needs to be correspondingly added to the smart contract compiler, so the smart contract compiler also writes the processing logic corresponding to the function instruction into the service smart contract.

[0119] FIG. 5b is a schematic diagram illustrating deployment of a BASE64 encoding/decoding operation corresponding to Embodiment 2 and Embodiment 4, according to some embodiments of the present specification. As shown in FIG. 5b, first, at least one of the BASE64 encoding instruction or the BASE64 decoding instruction is added to the instruction set of the blockchain virtual machine, and at least one of the BASE64 encoding logic or the BASE64 decoding logic is deployed in the blockchain virtual machine. Second, at least one of the BASE64 encoding instruction or the BASE64 decoding instruction is added to the instruction set of the smart contract compiler. Third, the service smart contract (which includes the contract identifier of the BASE64 encoding smart contract and the contract identifier of the BASE64 decoding smart contract) compiled by using the smart contract compiler is deployed in the blockchain network.

[0120] Embodiment 5

[0121] FIG. 6 is a schematic flowchart illustrating a blockchain smart contract-based signature verification method, according to some embodiments of the present specification. The method includes the following steps:

[0122] S600: A node in a blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0123] S602: When executing the service initiation transaction, each node in the blockchain network invokes a service smart contract by using a blockchain virtual machine.

[0124] S604: The node triggers execution of RSA signature verification logic based on an RSA signature verification instruction in the service smart contract by using the blockchain virtual machine, to perform an RSA signature verification operation on a service signature.

[0125] In some embodiments, the blockchain network needs to be pre-configured, so the following can be implemented:

[0126] (1) An instruction set of the blockchain virtual machine includes the RSA signature verification instruction, and the RSA signature verification logic corresponding to the RSA signature verification instruction is deployed in the blockchain virtual machine.

[0127] (2) An instruction set of a smart contract compiler includes the RSA signature verification instruction, and the service smart contract compiled by using the smart contract compiler includes the RSA signature verification instruction.

[0128] (3) The service smart contract is deployed in the blockchain network.

[0129] The service smart contract is a smart contract corresponding to a service that needs to invoke an RSA signature verification function.

[0130] In Embodiment 5, after invoking the service smart contract, the blockchain virtual machine reads bytecode or binary code in the service smart contract, and executes the bytecode or the binary code. When the blockchain virtual machine reads the RSA signature verification instruction in the service smart contract, it is equivalent to determining that an RSA signature verification operation needs to be performed. Therefore, the blockchain virtual machine triggers execution of the locally pre-deployed RSA signature verification logic, to perform a signature verification operation on the service signature.

[0131] In practice, depending on a specific service demand, the following cases exist:

[0132] (1) The service initiation transaction includes the service signature, signed data corresponding to the service signature, and a public key used to verify the service signature.

[0133] (2) The service initiation transaction includes the service signature, an abstract of signed data corresponding to the service signature, and a public key used to verify the service signature.

[0134] (3) The service initiation transaction includes the service signature and signed data corresponding to the service signature, and the service smart contract includes a public key used to verify the service signature.

[0135] (4) The service initiation transaction includes the service signature and an abstract of signed data corresponding to the service signature, and the service smart contract includes a public key used to verify the service signature.

[0136] It is worthwhile to note that if the blockchain virtual machine triggers execution of the RSA signature verification logic based on the RSA signature verification instruction in the service smart contract, a stack-based parameter passing

method is usually used in an execution process for data transfer.

[0137] In addition, the blockchain virtual machine can also execute the RSA signature verification logic by using a memory-based parameter passing method (a pre-compiled contract method), and the method is described in the following Embodiment 6.

[0138] Embodiment 6

[0139] FIG. 7 is a schematic flowchart illustrating a blockchain smart contract-based signature verification method, according to some embodiments of the present specification. The method includes the following steps:

[0140] S700: A node in a blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0141] S702: When executing the service initiation transaction, each node in the blockchain network invokes a service smart contract by using a blockchain virtual machine.

[0142] S704: The node invokes an RSA signature verification smart contract based on a contract identifier of the RSA signature verification smart contract in the service smart contract by using the blockchain virtual machine.

[0143] S706: The node triggers execution of RSA signature verification logic based on an RSA signature verification instruction in the RSA signature verification smart contract by using the blockchain virtual machine, to perform an RSA signature verification operation on a service signature.

[0144] In some embodiments, the blockchain network needs to be pre-configured, so the following can be implemented:

[0145] (1) An instruction set of the blockchain virtual machine includes the RSA signature verification instruction, and the RSA signature verification logic corresponding to the RSA signature verification instruction is deployed in the blockchain virtual machine.

[0146] (2) An instruction set of the smart contract compiler includes the RSA signature verification instruction, the service smart contract compiled by using the smart contract compiler includes the contract identifier of the RSA signature verification smart contract, and the RSA signature verification smart contract is a smart contract pre-deployed in the blockchain network.

[0147] (3) The service smart contract is deployed in the blockchain network.

[0148] A difference between Embodiment 6 and Embodiment 5 mainly lies in the

following: In Embodiment 6, after the blockchain virtual machine invokes the service smart contract, when the blockchain virtual machine reads the contract identifier of the RSA signature verification smart contract, it is equivalent to determining that the RSA signature verification smart contract needs to be further invoked. The blockchain virtual machine invokes the RSA signature verification smart contract, and also reads bytecode or binary code in the RSA signature verification smart contract. When the blockchain virtual machine reads the RSA signature verification instruction, it is equivalent to determining that an RSA signature verification operation needs to be performed. Therefore, the blockchain virtual machine triggers execution of the locally pre-deployed RSA signature verification logic, to perform an RSA signature verification operation on the service signature.

[0149] That is, in Embodiment 6, when the smart contract compiler compiles the service smart contract, if the smart contract compiler identifies that invoking of the RSA signature verification operation is declared in the service smart contract, the smart contract compiler compiles the declaration into the contract identifier of the RSA signature verification smart contract instead of the RSA signature verification instruction. As such, when invoking the service smart contract, the blockchain virtual machine further invokes the RSA signature verification smart contract.

[0150] FIG. 8a is a schematic diagram illustrating deployment of an RSA signature verification operation corresponding to Embodiment 5, according to some embodiments of the present specification. As shown in FIG. 8a, first, the RSA signature verification instruction is added to the instruction set of the blockchain virtual machine, and the RSA signature verification logic is deployed in the blockchain virtual machine. Second, the RSA signature verification instruction is added to the instruction set of the smart contract compiler. Third, the service smart contract (which includes the RSA signature verification instruction) compiled by using the smart contract compiler is deployed in the blockchain network.

[0151] FIG. 8b is a schematic diagram illustrating deployment of an RSA signature verification operation corresponding to Embodiment 6, according to some embodiments of the present specification. As shown in FIG. 8b, first, the RSA signature verification instruction is added to the instruction set of the blockchain virtual machine, and the RSA signature verification logic is deployed in the blockchain virtual machine. Second, the RSA signature verification instruction is added to the instruction set of the smart contract compiler. Third, the service smart contract (which includes the contract

identifier of the RSA signature verification smart contract) compiled by using the smart contract compiler is deployed in the blockchain network.

[0152] Embodiment 7

[0153] FIG. 9 is a schematic flowchart illustrating a blockchain smart contract-based data processing method, according to some embodiments of the present specification. The method includes the following steps:

[0154] S900: A node in a blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0155] S902: When executing the service initiation transaction, each node in the blockchain network invokes a service smart contract by using a blockchain virtual machine.

[0156] S904: The node triggers execution of JSON processing logic based on a JSON processing instruction in the service smart contract by using the blockchain virtual machine, to perform a JSON processing operation on data to be processed.

[0157] In some embodiments, the blockchain network needs to be pre-configured, so the following can be implemented:

[0158] (1) An instruction set of the blockchain virtual machine includes the JSON processing instruction, and the JSON processing logic corresponding to the JSON processing instruction is deployed in the blockchain virtual machine.

[0159] (2) An instruction set of a smart contract compiler includes the JSON processing instruction, and the service smart contract compiled by using the smart contract compiler includes the JSON processing instruction.

[0160] (3) The service smart contract is deployed in the blockchain network.

[0161] The service smart contract is a smart contract corresponding to a service that needs to invoke a JSON processing function. JSON processing specifically includes JSON data parsing and JSON data generation.

[0162] In Embodiment 7, after invoking the service smart contract, the blockchain virtual machine reads bytecode or binary code in the service smart contract, and executes the bytecode or the binary code. When the blockchain virtual machine reads the JSON processing instruction in the service smart contract, it is equivalent to determining that a JSON processing operation needs to be performed. Therefore, the blockchain virtual machine triggers execution of the locally pre-deployed JSON processing logic, to perform a JSON processing operation on the data to be processed.

[0163] In practice, depending on a specific service demand, the data to be processed

can be included in the service initiation transaction, can be included in the service smart contract, or can be generated when the blockchain virtual machine executes the service initiation transaction.

[0164] In addition, it is worthwhile to note that if the blockchain virtual machine triggers execution of the JSON processing logic based on the JSON processing instruction in the service smart contract, a stack-based parameter passing method (that is, data obtained after JSON processing is generally written into a stack) is usually used in an execution process for data transfer. However, a data length of the data obtained after JSON processing is not fixed, and if the stack-based parameter passing method is used, implementation is usually relatively complex.

[0165] Therefore, the following Embodiment 8 provides another blockchain smart contract-based data processing method (a pre-compiled contract method).

[0166] Embodiment 8

[0167] FIG. 10 is a schematic flowchart illustrating a blockchain smart contract-based data processing method, according to some embodiments of the present specification. The method includes the following steps:

[0168] S1000: A node in a blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0169] S1002: When executing the service initiation transaction, each node in the blockchain network invokes a service smart contract by using a blockchain virtual machine.

[0170] S1004: The node invokes a JSON processing smart contract based on a contract identifier of the JSON processing smart contract in the service smart contract by using the blockchain virtual machine.

[0171] S1006: The node triggers execution of JSON processing logic based on a JSON processing instruction in the JSON processing smart contract by using the blockchain virtual machine, to perform a JSON processing operation on data to be processed.

[0172] In some embodiments, the blockchain network needs to be pre-configured, so the following can be implemented:

[0173] (1) An instruction set of the blockchain virtual machine includes the JSON processing instruction, and the JSON processing logic corresponding to the JSON processing instruction is deployed in the blockchain virtual machine.

[0174] (2) An instruction set of a smart contract compiler includes the JSON

processing instruction, the service smart contract compiled by using the smart contract compiler includes the contract identifier of the JSON processing smart contract, and the JSON processing smart contract is a smart contract pre-deployed in the blockchain network.

[0175] (3) The service smart contract is deployed in the blockchain network.

[0176] A difference between Embodiment 8 and Embodiment 7 mainly lies in the following: In Embodiment 8, after the blockchain virtual machine invokes the service smart contract, when the blockchain virtual machine reads the contract identifier of the JSON processing smart contract, it is equivalent to determining that the JSON processing smart contract needs to be further invoked. The blockchain virtual machine invokes the JSON processing smart contract, and also reads bytecode or binary code in the JSON processing smart contract. When the blockchain virtual machine reads the JSON processing instruction, it is equivalent to determining that a JSON processing operation needs to be performed. Therefore, the blockchain virtual machine triggers execution of the locally pre-deployed JSON processing logic, to perform a JSON processing operation on the data to be processed.

[0177] That is, in Embodiment 8, when the smart contract compiler compiles the service smart contract, if the smart contract compiler identifies that invoking of the JSON processing operation is declared in the service smart contract, the smart contract compiler compiles the declaration into the contract identifier of the JSON processing smart contract instead of the JSON processing instruction. As such, when invoking the service smart contract, the blockchain virtual machine further invokes the JSON processing smart contract.

[0178] FIG. 11a is a schematic diagram illustrating deployment of a JSON processing operation corresponding to Embodiment 7, according to some embodiments of the present specification. As shown in FIG. 11a, first, the JSON processing instruction is added to the instruction set of the blockchain virtual machine, and the JSON processing logic is deployed in the blockchain virtual machine. Second, the JSON processing instruction is added to the instruction set of the smart contract compiler. Third, the service smart contract (which includes the JSON processing instruction) compiled by using the smart contract compiler is deployed in the blockchain network.

[0179] FIG. 11b is a schematic diagram illustrating deployment of a JSON processing operation corresponding to Embodiment 8, according to some embodiments

of the present specification. As shown in FIG. 11b, first, the JSON processing instruction is added to the instruction set of the blockchain virtual machine, and the JSON processing logic is deployed in the blockchain virtual machine. Second, the JSON processing instruction is added to the instruction set of the smart contract compiler. Third, the service smart contract (which includes the contract identifier of the JSON processing smart contract) compiled by using the smart contract compiler is deployed in the blockchain network.

[0180] Embodiment 9

[0181] FIG. 12 is a schematic flowchart illustrating a blockchain smart contract-based data processing method, according to some embodiments of the present specification. The method includes the following steps:

[0182] S1200: A node in a blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0183] S1202: When executing the service initiation transaction, each node in the blockchain network invokes a service smart contract by using a blockchain virtual machine.

[0184] S1204: The node triggers execution of XML processing logic based on an XML processing instruction in the service smart contract by using the blockchain virtual machine, to perform an XML processing operation on data to be processed.

[0185] In some embodiments, the blockchain network needs to be pre-configured, so the following can be implemented:

[0186] (1) An instruction set of the blockchain virtual machine includes the XML processing instruction, and the XML processing logic corresponding to the XML processing instruction is deployed in the blockchain virtual machine.

[0187] (2) An instruction set of a smart contract compiler includes the XML processing instruction, and the service smart contract compiled by using the smart contract compiler includes the XML processing instruction.

[0188] (3) The service smart contract is deployed in the blockchain network.

[0189] The service smart contract is a smart contract corresponding to a service that needs to invoke an XML processing function. XML processing specifically includes XML data parsing and XML data generation.

[0190] In Embodiment 9, after invoking the service smart contract, the blockchain virtual machine reads bytecode or binary code in the service smart contract, and executes the bytecode or the binary code. When the blockchain virtual machine reads

the XML processing instruction in the service smart contract, it is equivalent to determining that an XML processing operation needs to be performed. Therefore, the blockchain virtual machine triggers execution of the locally pre-deployed XML processing logic, to perform an XML processing operation on the data to be processed.

[0191] In practice, depending on a specific service demand, the data to be processed can be included in the service initiation transaction, can be included in the service smart contract, or can be generated when the blockchain virtual machine executes the service initiation transaction.

[0192] In addition, it is worthwhile to note that if the blockchain virtual machine triggers execution of the XML processing logic based on the XML processing instruction in the service smart contract, a stack-based parameter passing method (that is, data obtained after XML processing is generally written into a stack) is usually used in an execution process for data transfer. However, a data length of the data obtained after XML processing is not fixed, and if the stack-based parameter passing method is used, implementation is usually relatively complex.

[0193] Therefore, the following Embodiment 10 provides another blockchain smart contract-based data processing method (a pre-compiled contract method).

[0194] Embodiment 10

[0195] FIG. 13 is a schematic flowchart illustrating a blockchain smart contract-based data processing method, according to some embodiments of the present specification. The method includes the following steps:

[0196] S1300: A node in a blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0197] S1302: When executing the service initiation transaction, each node in the blockchain network invokes a service smart contract by using a blockchain virtual machine.

[0198] S1304: The node invokes an XML processing smart contract based on a contract identifier of the XML processing smart contract in the service smart contract by using the blockchain virtual machine.

[0199] S1306: The node triggers execution of XML processing logic based on an XML processing instruction in the XML processing smart contract by using the blockchain virtual machine, to perform an XML processing operation on data to be processed.

[0200] In some embodiments, the blockchain network needs to be pre-configured,

so the following can be implemented:

[0201] (1) An instruction set of the blockchain virtual machine includes the XML processing instruction, and the XML processing logic corresponding to the XML processing instruction is deployed in the blockchain virtual machine.

[0202] (2) An instruction set of a smart contract compiler includes the XML processing instruction, the service smart contract compiled by using the smart contract compiler includes the contract identifier of the XML processing smart contract, and the XML processing smart contract is a smart contract pre-deployed in the blockchain network.

[0203] (3) The service smart contract is deployed in the blockchain network.

[0204] A difference between Embodiment 10 and Embodiment 9 mainly lies in the following: In Embodiment 10, after the blockchain virtual machine invokes the service smart contract, when the blockchain virtual machine reads the contract identifier of the XML processing smart contract, it is equivalent to determining that the XML processing smart contract needs to be further invoked. The blockchain virtual machine invokes the XML processing smart contract, and also reads bytecode or binary code in the XML processing smart contract. When the blockchain virtual machine reads the XML processing instruction, it is equivalent to determining that an XML processing operation needs to be performed. Therefore, the blockchain virtual machine triggers execution of the locally pre-deployed XML processing logic, to perform an XML processing operation on the data to be processed.

[0205] That is, in Embodiment 10, when the smart contract compiler compiles the service smart contract, if the smart contract compiler identifies that invoking of the XML processing operation is declared in the service smart contract, the smart contract compiler compiles the declaration into the contract identifier of the XML processing smart contract instead of the XML processing instruction. As such, when invoking the service smart contract, the blockchain virtual machine further invokes the XML processing smart contract.

[0206] FIG. 14a is a schematic diagram illustrating deployment of an XML processing operation corresponding to Embodiment 9, according to some embodiments of the present specification. As shown in FIG. 14a, first, the XML processing instruction is added to the instruction set of the blockchain virtual machine, and the XML processing logic is deployed in the blockchain virtual machine. Second, the XML processing instruction is added to the instruction set of the smart contract compiler.

Third, the service smart contract (which includes the XML processing instruction) compiled by using the smart contract compiler is deployed in the blockchain network.

[0207] FIG. 14b is a schematic diagram illustrating deployment of an XML processing operation corresponding to Embodiment 10, according to some embodiments of the present specification. As shown in FIG. 14b, first, the XML processing instruction is added to the instruction set of the blockchain virtual machine, and the XML processing logic is deployed in the blockchain virtual machine. Second, the XML processing instruction is added to the instruction set of the smart contract compiler. Third, the service smart contract (which includes the contract identifier of the XML processing smart contract) compiled by using the smart contract compiler is deployed in the blockchain network.

[0208] Embodiment 11

[0209] FIG. 15 is a schematic flowchart illustrating a blockchain smart contract-based transaction hash acquisition method, according to some embodiments of the present specification. The method includes the following steps:

[0210] S1500: A node in a blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0211] S1502: When executing the service initiation transaction, each node in the blockchain network invokes a service smart contract by using a blockchain virtual machine.

[0212] S1504: The node triggers execution of transaction hash acquisition logic based on a transaction hash acquisition instruction in the service smart contract by using the blockchain virtual machine, to obtain a transaction hash of the service initiation transaction.

[0213] In some embodiments, the blockchain network needs to be pre-configured, so the following can be implemented:

[0214] (1) An instruction set of the blockchain virtual machine includes a transaction hash acquisition instruction, and transaction hash acquisition logic corresponding to the transaction hash acquisition instruction is deployed in the blockchain virtual machine.

[0215] (2) An instruction set of a smart contract compiler includes the transaction hash acquisition instruction, and the service smart contract compiled by using the smart contract compiler includes the transaction hash acquisition instruction.

[0216] (3) The service smart contract is deployed in the blockchain network.

[0217] The service smart contract is a smart contract corresponding to a service that

needs to invoke a transaction hash acquisition function.

[0218] In Embodiment 11, after invoking the service smart contract, the blockchain virtual machine reads bytecode or binary code in the service smart contract, and executes the bytecode or the binary code. When the blockchain virtual machine reads the transaction hash acquisition instruction in the service smart contract, it is equivalent to determining that a transaction hash acquisition operation needs to be performed. Therefore, the blockchain virtual machine triggers execution of the locally pre-deployed transaction hash acquisition logic, to perform a transaction hash acquisition operation on data to be processed.

[0219] In practice, before invoking the service smart contract, each node in the blockchain network initializes a context of the blockchain virtual machine, and writes the transaction hash of the service initiation transaction into the context.

[0220] Thus, in step S1504, specifically, the node triggers execution of the transaction hash acquisition logic based on the transaction hash acquisition instruction in the service smart contract by using the blockchain virtual machine, to obtain the transaction hash of the service initiation transaction from the context.

[0221] In addition, it is worthwhile to note that if the blockchain virtual machine triggers execution of the transaction hash acquisition logic based on the transaction hash acquisition instruction in the service smart contract, a stack-based parameter passing method is generally used in an execution process for data transfer. The following Embodiment 12 provides another blockchain smart contract-based transaction hash acquisition method (a pre-compiled contract method).

[0222] Embodiment 12

[0223] FIG. 16 is a schematic flowchart illustrating a blockchain smart contract-based transaction hash acquisition method, according to some embodiments of the present specification. The method includes the following steps:

[0224] S1600: A node in a blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0225] S1602: When executing the service initiation transaction, each node in the blockchain network invokes a service smart contract by using a blockchain virtual machine.

[0226] S1604: The node invokes a transaction hash acquisition smart contract based on a contract identifier of the transaction hash acquisition smart contract in the service smart contract by using the blockchain virtual machine.

[0227] S1606: The node triggers execution of transaction hash acquisition logic based on a transaction hash acquisition instruction in the transaction hash acquisition smart contract by using the blockchain virtual machine, to obtain a transaction hash of the service initiation transaction.

[0228] In some embodiments, the blockchain network needs to be pre-configured, so the following can be implemented:

[0229] (1) An instruction set of the blockchain virtual machine includes a transaction hash acquisition instruction, and transaction hash acquisition logic corresponding to the transaction hash acquisition instruction is deployed in the blockchain virtual machine.

[0230] (2) An instruction set of a smart contract compiler includes the transaction hash acquisition instruction, the service smart contract compiled by using the smart contract compiler includes the contract identifier of the transaction hash acquisition smart contract, and the transaction hash acquisition smart contract is a smart contract pre-deployed in the blockchain network.

[0231] (3) The service smart contract is deployed in the blockchain network.

[0232] A difference between Embodiment 12 and Embodiment 11 mainly lies in the following: In Embodiment 12, after the blockchain virtual machine invokes the service smart contract, when the blockchain virtual machine reads the contract identifier of the transaction hash acquisition smart contract, it is equivalent to determining that the transaction hash acquisition smart contract needs to be further invoked. The blockchain virtual machine invokes the transaction hash acquisition smart contract, and also reads bytecode or binary code in the transaction hash acquisition smart contract. When the blockchain virtual machine reads the transaction hash acquisition instruction, it is equivalent to determining that a transaction hash acquisition operation needs to be performed. Therefore, the blockchain virtual machine triggers execution of the locally pre-deployed transaction hash acquisition logic, to obtain the transaction hash of the service initiation transaction.

[0233] That is, in Embodiment 12, when the smart contract compiler compiles the service smart contract, if the smart contract compiler identifies that invoking of the transaction hash acquisition operation is declared in the service smart contract, the smart contract compiler compiles the declaration into the contract identifier of the transaction hash acquisition smart contract instead of the transaction hash acquisition instruction. As such, when invoking the service smart contract, the blockchain virtual machine further invokes the transaction hash acquisition smart contract.

[0234] In practice, before invoking the service smart contract, each node in the blockchain network initializes a context of the blockchain virtual machine, and writes the transaction hash of the service initiation transaction into the context.

[0235] Thus, in step S1606, specifically, the node triggers execution of the transaction hash acquisition logic based on transaction hash acquisition instruction in the transaction hash acquisition smart contract by using the blockchain virtual machine, to obtain the transaction hash of the service initiation transaction from the context.

[0236] FIG. 17a is a schematic diagram illustrating deployment of a transaction hash acquisition operation corresponding to Embodiment 11, according to some embodiments of the present specification. As shown in FIG. 17a, first, the transaction hash acquisition instruction is added to the instruction set of the blockchain virtual machine, and the transaction hash acquisition logic is deployed in the blockchain virtual machine. Second, the transaction hash acquisition instruction is added to the instruction set of the smart contract compiler. Third, the service smart contract (which includes the transaction hash acquisition instruction) compiled by using the smart contract compiler is deployed in the blockchain network.

[0237] FIG. 17b is a schematic diagram illustrating deployment of a transaction hash acquisition operation corresponding to Embodiment 12, according to some embodiments of the present specification. As shown in FIG. 17b, first, the transaction hash acquisition instruction is added to the instruction set of the blockchain virtual machine, and the transaction hash acquisition logic is deployed in the blockchain virtual machine. Second, the transaction hash acquisition instruction is added to the instruction set of the smart contract compiler. Third, the service smart contract (which includes the contract identifier of the transaction hash acquisition smart contract) compiled by using the smart contract compiler is deployed in the blockchain network.

[0238] Embodiment 13

[0239] FIG. 18 is a schematic flowchart illustrating a blockchain smart contract-based transfer method, according to some embodiments of the present specification. The method includes the following steps:

[0240] S1800: A node in a blockchain network obtains a transfer transaction by using an external transfer account, and broadcasts the transfer transaction to other nodes.

[0241] That is, the node receives the transfer transaction constructed and sent by a client device (which logs in to the external transfer account).

[0242] S1802: When executing the transfer transaction, each node in the blockchain

network invokes a service smart contract by using a blockchain virtual machine, and triggers execution of input/output verification logic based on an input/output verification instruction in the service smart contract.

[0243] S1804: If the verification result is affirmative, the node executes user asset table modification logic in the service smart contract by using the blockchain virtual machine.

[0244] In an Ethereum architecture, external accounts are in a one-to-one correspondence with users. A transfer behavior between two users is actually a transfer behavior between two external accounts.

[0245] Assume that external account A expects to transfer money to external account B. External account A first constructs a transfer transaction that includes a transfer amount and an account address of external account B, and then the transfer transaction is broadcast to each node in the blockchain network. After successfully verifying the transfer transaction, each node executes the transfer transaction, and writes the transfer amount into external account B.

[0246] It is worthwhile to note that the verification event of each node for the transfer transaction generally includes verifying whether balance of external account A is sufficient, that is, whether the balance of external account A is greater than or equal to the transfer amount. If affirmative, the verification succeeds. It means that in the existing Ethereum architecture, if money needs to be transferred, account balance of a transferor and a transfer amount are exposed to each node in the blockchain network.

[0247] In Embodiment 13, service demands are actually provided, that is, for a transfer transaction between nodes in the blockchain network, whether asset balance of a transfer user is sufficient to cover a transfer amount is determined without exposing the asset balance of the transfer user and the transfer amount.

[0248] To satisfy the service demand, a new transfer model is used in the present application. Instead of using the balance in the external account in the blockchain network for transfer and collection, a service smart contract is created, and an asset system of a user is re-established in the service smart contract. In the asset system, there is no longer a concept of the asset balance. An asset owned by each user is equivalent to a virtual object. The user spends an asset, and the asset disappears.

[0249] Specifically, the service smart contract is deployed in the blockchain network, the service smart contract corresponds to a user asset table, and the user asset table is used to record an asset corresponding to each external account (external accounts are

in a one-to-one correspondence with users). Any asset is data that includes an encrypted amount, and the encrypted amount is obtained after an amount of the asset is encrypted.

[0250] The following Table 3 provides an example of the user asset table.

External account 1	Asset 1, and asset 2
External account 2	Asset 3, asset 4, and asset 5
External account 3	Asset 6

Table 3

[0251] For example, external account 1 transfers 120 yuan to external account 2. Assume that an amount of asset 1 is 100 yuan and an amount of asset 2 is 50 yuan. External account 1 needs to spend asset 1 and asset 2, and then asset 1 and assert 2 disappear. As such, asset 7 (an amount of 120 yuan) and asset 8 (an amount of 30 yuan) are generated, external account 1 obtains asset 8, and external account 2 obtains asset 7. It can be understood that asset 1 + asset 2 is actually input of the transfer, and asset 7 + asset 8 is actually output of the transfer. After the transfer, Table 3 is updated to Table 4 shown below.

External account 1	Asset 8
External account 2	Asset 3, asset 4, asset 5, and asset 7
External account 3	Asset 6

Table 4

[0252] Therefore, in the present application, for a transfer transaction, a problem of verifying the balance is converted into a problem of verifying whether balance is kept between the input and the output. Because an amount of each asset is actually encrypted and is not exposed to the blockchain virtual machine, it means that when executing a transfer transaction, the blockchain virtual machine needs to verify whether an input asset is equal to an output asset by using a homomorphic encryption algorithm (such as the Pedersen commitment algorithm). If the verification result is affirmative, it means that the transfer transaction is feasible (which is equivalent to indicating that balance of a transferor is sufficient).

[0253] For a specific implementation, in Embodiment 13, the blockchain network needs to be pre-configured, so the following can be implemented:

[0254] (1) An instruction set of the blockchain virtual machine includes the input/output verification instruction, and the input/output verification logic corresponding to the input/output verification instruction is deployed in the blockchain

virtual machine.

[0255] The input/output verification logic includes: verifying, by using the homomorphic encryption algorithm for any transfer transaction, whether the sum of an amount of a transfer asset specified in the transfer transaction and an amount of a change asset specified in the transfer transaction is equal to the sum of amounts of expense assets specified in the transfer transaction.

[0256] It is further worthwhile to note that the transfer transaction is actually the aforementioned service initiation transaction, but a service here is specifically an encrypted transfer service.

[0257] (2) An instruction set of a smart contract compiler includes the input/output verification instruction, and the service smart contract compiled by using the smart contract compiler includes the input/output verification instruction.

[0258] (3) The service smart contract is deployed in the blockchain network.

[0259] The service smart contract is a smart contract for hiding the transfer amount and the asset balance.

[0260] In step S1800, the node is actually a node corresponding to a transfer user. The external transfer account is actually an external account controlled by the transfer user.

[0261] That the node obtains the transfer transaction by using the external transfer account is actually that the transfer user logs in to the client device (or "wallet"), constructs the transfer transaction, and sends the transfer transaction to other nodes.

[0262] The transfer transaction includes a transfer asset, a change asset, and at least one expense asset. The at least one expense asset is actually an asset that is input into the transfer, and the transfer asset and the change asset are actually assets that are output from the transfer.

[0263] In Embodiment 13, after invoking the service smart contract, the blockchain virtual machine reads bytecode or binary code in the service smart contract, and executes the bytecode or the binary code. When the blockchain virtual machine reads the input/output verification instruction in the service smart contract, it is equivalent to determining that an input/output verification operation needs to be performed. Therefore, the blockchain virtual machine triggers execution of the locally pre-deployed input/output verification logic, to verify, by using the homomorphic encryption algorithm, whether the sum of the amount of the transfer asset and the amount of the change asset is equal to the sum of the amounts of the expense assets.

[0264] If the verification result is affirmative, it indicates that the transfer transaction is feasible. Thus, the blockchain virtual machine needs to execute the transfer transaction, that is, modify the user asset table based on the transfer transaction. The user asset table modification logic for modifying the user asset table is actually written into the service smart contract in advance. The user asset table modification logic is to release a correspondence between the external transfer account and each expense asset, establish a correspondence between the external transfer account and the change asset, and establish a correspondence between an external collection account and the transfer asset.

[0265] In addition, in practice, the following cases may occur: When initiating a transfer transaction, some malicious users set an actual amount of a transfer asset included in the transfer transaction to a negative value, and steal an asset of another person by using a transfer method; and when initiating a transfer transaction, a user performs an unintended operation to set an actual amount of a change asset included in the transfer transaction to a negative value, and consequently, an amount of an asset that is actually transferred by the user is greater than an amount of a transfer asset. To prevent the aforementioned cases from occurring, a zero-knowledge proof method can be used, so when both the actual amount of the transfer asset and the actual amount of the change asset are invisible, it is verified whether both the actual amount of the transfer asset and the actual amount of the change asset fall into a specified value range, such as $(0, 2^{64}]$. If the verification result is affirmative, the possibility that the aforementioned cases occur is ruled out.

[0266] Specifically, in Embodiment 13, the blockchain network can be further preconfigured, so the following can be implemented:

[0267] (4) The instruction set of the blockchain virtual machine further includes an amount verification instruction, and amount verification logic corresponding to the amount verification instruction is further deployed in the blockchain virtual machine.

[0268] (5) The instruction set of the smart contract compiler further includes the amount verification instruction, and the service smart contract compiled by using the smart contract compiler further includes the amount verification instruction.

[0269] Based on this, the transfer transaction initiated by the user further needs to include proof-related data (such as Bullet Proof or a Borromean ring signature) used to implement zero-knowledge proof.

[0270] When reading the amount verification instruction in the service smart

contract, the blockchain virtual machine triggers execution of the locally pre-deployed amount verification logic, that is, verifies, based on the proof-related data, whether both the amount of the transfer asset and the amount of the change asset fall into the specified value range.

[0271] In addition, in Embodiment 13, the service smart contract can further include an expense asset verification logic, so when executing the service smart contract, the blockchain virtual machine can further verify, based on the user asset table, whether assets corresponding to the external transfer account include the expense assets. If the verification result is affirmative, it indicates that an expense asset to be used by the transfer user is an asset owned by the transfer user.

[0272] In addition, a data structure of any asset can further include a first status parameter that indicates that the asset has been spent or remains unspent. The service smart contract can further include first status parameter-related logic, so when executing the service smart contract, the blockchain virtual machine verifies, for each expense asset in the transfer transaction based on a first status parameter included in the expense asset, whether the expense asset remains unspent. In addition, the first status parameter-related logic further includes: if all final verification results for the transfer transaction are affirmative, modifying the first status parameter included in the expense asset, so the modified first status parameter represents that the expense asset has been spent. If any result in all the final verification results for the transfer transaction is no, the transfer transaction will fail, no expense asset is spent, and therefore, there is no need to modify the first status parameter.

[0273] Further, a data structure of any asset can further include a second status parameter that represents that the asset exists or does not exist. The service smart contract can further include second status parameter-related logic, so when executing the service smart contract, the blockchain virtual machine verifies, for each expense asset in the transfer transaction based on the second status parameter included in the expense asset, whether the expense asset exists. As such, a malicious user can be effectively prevented from faking an asset that does not exist and using the faked asset for transfer.

[0274] In addition, it is worthwhile to note that if the blockchain virtual machine triggers execution of the input/output verification logic and the amount verification logic based on the input/output verification instruction and the amount verification instruction in the service smart contract, a stack-based parameter passing method is

generally used in an execution process for data transfer. The following Embodiment 14 provides another blockchain smart contract-based transfer method (a pre-compiled contract method).

[0275] Embodiment 14

[0276] FIG. 19 is a schematic flowchart illustrating a blockchain smart contract-based transaction hash acquisition method, according to some embodiments of the present specification. The method includes the following steps:

[0277] S1900: A node in a blockchain network obtains a transfer transaction by using an external transfer account, and broadcasts the transfer transaction to other nodes.

[0278] S1902: When executing the transfer transaction, each node in the blockchain network invokes a service smart contract by using a blockchain virtual machine, and invokes an input/output verification smart contract based on a contract identifier of the input/output verification smart contract in the service smart contract.

[0279] S1904: The node triggers execution of input/output verification logic based on an input/output verification instruction in the input/output verification smart contract.

[0280] S1906: If the verification result is affirmative, the node executes user asset table modification logic in the service smart contract by using the blockchain virtual machine.

[0281] In Embodiment 14, the blockchain network needs to be pre-configured, so the following can be implemented:

[0282] (1) An instruction set of the blockchain virtual machine includes the input/output verification instruction, and input/output verification logic corresponding to the input/output verification instruction is deployed in the blockchain virtual machine.

[0283] (2) An instruction set of a smart contract compiler includes the input/output verification instruction, the service smart contract compiled by using the smart contract compiler includes the contract identifier of the input/output verification smart contract, and the input/output verification smart contract is a smart contract pre-deployed in the blockchain network.

[0284] (3) The service smart contract is deployed in the blockchain network.

[0285] A difference between Embodiment 14 and Embodiment 13 mainly lies in the following: In Embodiment 14, after the blockchain virtual machine invokes the service smart contract, when the blockchain virtual machine reads the contract identifier of the input/output verification smart contract, it is equivalent to determining that the input/output verification smart contract needs to be further invoked. The blockchain

virtual machine invokes the input/output verification smart contract, and also reads bytecode or binary code in the input/output verification smart contract. When the blockchain virtual machine reads the input/output verification instruction, it is equivalent to determining that an input/output verification operation needs to be performed. Therefore, the blockchain virtual machine triggers execution of the locally pre-deployed input/output verification logic.

[0286] That is, in Embodiment 14, when the smart contract compiler compiles the service smart contract, if the smart contract compiler identifies that invoking of the input/output verification operation is declared in the service smart contract, the smart contract compiler compiles the declaration into the contract identifier of the input/output verification smart contract instead of the input/output verification instruction. As such, when invoking the service smart contract, the blockchain virtual machine further invokes the input/output verification smart contract.

[0287] Further, in Embodiment 14, the blockchain network needs to be pre-configured, so the following can be implemented:

[0288] (4) The instruction set of the blockchain virtual machine further includes an amount verification instruction, and amount verification logic corresponding to the amount verification instruction is further deployed in the blockchain virtual machine.

[0289] (5) The instruction set of the smart contract compiler further includes the amount verification instruction, the service smart contract compiled by using the smart contract compiler further includes a contract identifier of an amount verification smart contract, and the amount verification smart contract is a smart contract pre-deployed in the blockchain network.

[0290] As such, in Embodiment 14, each node in the blockchain network invokes the amount verification smart contract based on the contract identifier of the amount verification smart contract in the service smart contract by using the blockchain virtual machine. The node triggers execution of the amount verification logic based on the amount verification instruction in the amount verification smart contract, to verify, based on proof-related data, whether both an amount of a transfer asset and an amount of a change asset fall into a specified value range.

[0291] In addition, it is worthwhile to note that, in Embodiment 14, an executed verification event is consistent with that in Embodiment 13. However, input/output verification and amount verification are implemented by invoking a pre-compiled contract (namely, the input/output verification smart contract and the amount

verification smart contract).

[0292] In Embodiment 13 and Embodiment 14, when there are a plurality of verification events performed for the transfer transaction, only when all verification results are affirmative, the blockchain virtual machine executes the user asset table modification logic.

[0293] FIG. 20a is a schematic diagram illustrating deployment of a verification operation related to a transfer transaction corresponding to Embodiment 13, according to some embodiments of the present specification. As shown in FIG. 20a, first, the input/output verification instruction and the amount verification instruction are added to the instruction set of the blockchain virtual machine, and the input/output verification logic and the amount verification logic are deployed in the blockchain virtual machine. Second, the input/output verification instruction and the amount verification instruction are added to the instruction set of the smart contract compiler. Third, the service smart contract (which includes the input/output verification instruction and the amount verification instruction) compiled by using the smart contract compiler is deployed in the blockchain network.

[0294] FIG. 20b is a schematic diagram illustrating deployment of a verification operation related to a transfer transaction corresponding to Embodiment 14, according to some embodiments of the present specification. As shown in FIG. 20b, first, the input/output verification instruction and the amount verification instruction are added to the instruction set of the blockchain virtual machine, and the input/output verification logic and the amount verification logic are deployed in the blockchain virtual machine. Second, the input/output verification instruction and the amount verification instruction are added to the instruction set of the smart contract compiler. Third, the service smart contract (which includes the contract identifier of the input/output verification smart contract and the contract identifier of the amount verification smart contract) compiled by using the smart contract compiler is deployed in the blockchain network.

[0295] Embodiment 1 to Embodiment 14 are separately described above. In practice, the service initiation transaction can be specifically a transfer transaction. To implement the transfer transaction, related verification (for example, various verification mentioned in Embodiment 13) may need to be performed, and one or more of a BASE64 encoding operation, a BASE64 decoding operation, an RSA signature verification operation, a JSON processing operation, an XML processing operation, and a transaction hash acquisition operation further needs to be performed.

[0296] That is, one or more of the BASE64 encoding method, the BASE64 decoding method, the RSA signature verification method, the JSON processing method, the XML processing method, and the transaction hash acquisition method can be applied to the transfer method in Embodiment 13 or Embodiment 14. In this case, the aforementioned service initiation transaction is a transfer transaction.

[0297] In addition, it is worthwhile to note that in the embodiments of the present specification, if the service smart contract is not the only smart contract in the blockchain network, the service initiation transaction (and the transfer transaction) further needs to include a contract identifier of the service smart contract, so the blockchain virtual machine invokes the corresponding service smart contract based on the contract identifier included in the service initiation transaction (and the transfer transaction).

[0298] In addition, in the embodiments of the present specification, the instruction set of the blockchain virtual machine further includes at least one Ethereum instruction, and the Ethereum instruction is an instruction in an instruction set of the EVM. Relevant logic corresponding to each Ethereum instruction included in the instruction set of the blockchain virtual machine is deployed in the blockchain virtual machine. The instruction set of the smart contract compiler further includes at least one Ethereum instruction, and the Ethereum instruction is an instruction in the instruction set of the EVM. That is, the blockchain virtual machine in the present application can be obtained by expanding the EVM, and the smart contract compiler in the present application can be obtained by expanding the Ethereum smart contract compiler.

[0299] A blockchain system provided in the present specification includes a blockchain network.

[0300] An instruction set of a blockchain virtual machine includes an input/output verification instruction, and input/output verification logic corresponding to the input/output verification instruction is deployed in the blockchain virtual machine. An instruction set of a smart contract compiler includes the input/output verification instruction, and a service smart contract compiled by using the smart contract compiler includes the input/output verification instruction. The service smart contract is deployed in the blockchain network. The service smart contract corresponds to a user asset table, and the user asset table is used to record an asset corresponding to each external account. Any asset is data that includes an encrypted amount, and the encrypted amount is obtained after an amount of the asset is encrypted.

[0301] A node in the blockchain network obtains a transfer transaction by using an external transfer account, and broadcasts the transfer transaction to other nodes. The transfer transaction includes a transfer asset, a change asset, and at least one expense asset.

[0302] When executing the transfer transaction, each node in the blockchain network invokes the service smart contract by using the blockchain virtual machine, and triggers execution of the input/output verification logic based on the input/output verification instruction in the service smart contract, to verify, by using a homomorphic encryption algorithm, whether the sum of an amount of the transfer asset and an amount of the change asset is equal to the sum of amounts of expense assets. If the verification result is affirmative, the node executes user asset table modification logic in the service smart contract by using the blockchain virtual machine, to release a correspondence between the external transfer account and each expense asset, establish a correspondence between the external transfer account and the change asset, and establish a correspondence between an external collection account and the transfer asset.

[0303] A blockchain system provided in the present specification includes a blockchain network.

[0304] An instruction set of a blockchain virtual machine includes an input/output verification instruction, and input/output verification logic corresponding to the input/output verification instruction is deployed in the blockchain virtual machine. An instruction set of a smart contract compiler includes the input/output verification instruction, a service smart contract compiled by using the smart contract compiler includes a contract identifier of an input/output verification smart contract, and the input/output verification smart contract is a smart contract pre-deployed in the blockchain network. The service smart contract is deployed in the blockchain network. The service smart contract corresponds to a user asset table, and the user asset table is used to record an asset corresponding to each external account. Any asset is data that includes an encrypted amount, and the encrypted amount is obtained after an amount of the asset is encrypted.

[0305] A node in the blockchain network obtains a transfer transaction by using an external transfer account, and broadcasts the transfer transaction to other nodes. The transfer transaction includes a transfer asset, a change asset, and at least one expense asset.

[0306] When executing the transfer transaction, each node in the blockchain

network invokes the service smart contract by using the blockchain virtual machine, invokes the input/output verification smart contract based on the contract identifier of the input/output verification smart contract in the service smart contract, and triggers execution of the input/output verification logic based on the input/output verification instruction in the input/output verification smart contract, to verify, by using a homomorphic encryption algorithm, whether the sum of an amount of the transfer asset and an amount of the change asset is equal to the sum of amounts of expense assets. If the verification result is affirmative, the node executes user asset table modification logic in the service smart contract by using the blockchain virtual machine, to release a correspondence between the external transfer account and each expense asset, establish a correspondence between the external transfer account and the change asset, and establish a correspondence between an external collection account and the transfer asset.

[0307] A blockchain system provided in the present specification includes a blockchain network.

[0308] An instruction set of a blockchain virtual machine includes a BASE64 encoding instruction, and BASE64 encoding logic corresponding to the BASE64 encoding instruction is deployed in the blockchain virtual machine. An instruction set of a smart contract compiler includes the BASE64 encoding instruction, and a service smart contract compiled by using the smart contract compiler includes the BASE64 encoding instruction. The service smart contract is deployed in the blockchain network.

[0309] A node in the blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0310] When executing the service initiation transaction, each node in the blockchain network invokes the service smart contract by using the blockchain virtual machine, and triggers execution of the BASE64 encoding logic based on the BASE64 encoding instruction in the service smart contract by using the blockchain virtual machine, to perform an encoding operation on data to be encoded.

[0311] A blockchain system provided in the present specification includes a blockchain network.

[0312] An instruction set of a blockchain virtual machine includes a BASE64 encoding instruction, and BASE64 encoding logic corresponding to the BASE64 encoding instruction is deployed in the blockchain virtual machine. An instruction set of a smart contract compiler includes the BASE64 encoding instruction, a service smart contract compiled by using the smart contract compiler includes a contract identifier of

a BASE64 encoding smart contract, and the BASE64 encoding smart contract is a smart contract pre-deployed in the blockchain network. The service smart contract is deployed in the blockchain network.

[0313] A node in the blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0314] When executing the service initiation transaction, each node in the blockchain network invokes the service smart contract by using the blockchain virtual machine, invokes the BASE64 encoding smart contract based on the contract identifier of the BASE64 encoding smart contract in the service smart contract by using the blockchain virtual machine, and triggers execution of the BASE64 encoding logic based on the BASE64 encoding instruction in the BASE64 encoding smart contract by using the blockchain virtual machine, to perform an encoding operation on data to be encoded.

[0315] A blockchain system provided in the present specification includes a blockchain network.

[0316] An instruction set of a blockchain virtual machine includes a BASE64 decoding instruction, and BASE64 decoding logic corresponding to the BASE64 decoding instruction is deployed in the blockchain virtual machine. An instruction set of a smart contract compiler includes the BASE64 decoding instruction, and a service smart contract compiled by using the smart contract compiler includes the BASE64 decoding instruction. The service smart contract is deployed in the blockchain network.

[0317] A node in the blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0318] When executing the service initiation transaction, each node in the blockchain network invokes the service smart contract by using the blockchain virtual machine, and triggers execution of the BASE64 decoding logic based on the BASE64 decoding instruction in the service smart contract by using the blockchain virtual machine, to perform a decoding operation on data to be decoded.

[0319] A blockchain system provided in the present specification includes a blockchain network.

[0320] An instruction set of a blockchain virtual machine includes a BASE64 decoding instruction, and BASE64 decoding logic corresponding to the BASE64 decoding instruction is deployed in the blockchain virtual machine. An instruction set of a smart contract compiler includes the BASE64 decoding instruction, a service smart

contract compiled by using the smart contract compiler includes a contract identifier of a BASE64 decoding smart contract, and the BASE64 decoding smart contract is a smart contract pre-deployed in the blockchain network. The service smart contract is deployed in the blockchain network.

[0321] A node in the blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0322] When executing the service initiation transaction, each node in the blockchain network invokes the service smart contract by using the blockchain virtual machine, invokes the BASE64 decoding smart contract based on the contract identifier of the BASE64 decoding smart contract in the service smart contract by using the blockchain virtual machine, and triggers execution of the BASE64 decoding logic based on the BASE64 decoding instruction in the BASE64 decoding smart contract by using the blockchain virtual machine, to perform a decoding operation on data to be decoded.

[0323] A blockchain system provided in the present specification includes a blockchain network.

[0324] An instruction set of a blockchain virtual machine includes an RSA signature verification instruction, and RSA signature verification logic corresponding to the RSA signature verification instruction is deployed in the blockchain virtual machine. An instruction set of a smart contract compiler includes the RSA signature verification instruction, and a service smart contract compiled by using the smart contract compiler includes the RSA signature verification instruction. The service smart contract is deployed in the blockchain network.

[0325] A node in the blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0326] When executing the service initiation transaction, each node in the blockchain network invokes the service smart contract by using the blockchain virtual machine, and triggers execution of the RSA signature verification logic based on the RSA signature verification instruction in the service smart contract by using the blockchain virtual machine, to perform an RSA signature verification operation on a service signature.

[0327] A blockchain system provided in the present specification includes a blockchain network.

[0328] An instruction set of a blockchain virtual machine includes an RSA signature

verification instruction, and RSA signature verification logic corresponding to the RSA signature verification instruction is deployed in the blockchain virtual machine. An instruction set of a smart contract compiler includes the RSA signature verification instruction, a service smart contract compiled by using the smart contract compiler includes a contract identifier of an RSA signature verification smart contract, and the RSA signature verification smart contract is a smart contract pre-deployed in the blockchain network. The service smart contract is deployed in the blockchain network.

[0329] A node in the blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0330] When executing the service initiation transaction, each node in the blockchain network invokes the service smart contract by using the blockchain virtual machine, invokes the RSA signature verification smart contract based on the contract identifier of the RSA signature verification smart contract in the service smart contract by using the blockchain virtual machine, and triggers execution of the RSA signature verification logic based on the RSA signature verification instruction in the RSA signature verification smart contract by using the blockchain virtual machine, to perform an RSA signature verification operation on a service signature.

[0331] A blockchain system provided in the present specification includes a blockchain network.

[0332] An instruction set of a blockchain virtual machine includes a JSON processing instruction, and JSON processing logic corresponding to the JSON processing instruction is deployed in the blockchain virtual machine. An instruction set of a smart contract compiler includes the JSON processing instruction, and a service smart contract compiled by using the smart contract compiler includes the JSON processing instruction. The service smart contract is deployed in the blockchain network.

[0333] A node in the blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0334] When executing the service initiation transaction, each node in the blockchain network invokes the service smart contract by using the blockchain virtual machine, and triggers execution of the JSON processing logic based on the JSON processing instruction in the service smart contract by using the blockchain virtual machine, to perform a JSON processing operation on data to be processed.

[0335] A blockchain system provided in the present specification includes a

blockchain network.

[0336] An instruction set of a blockchain virtual machine includes a JSON processing instruction, and JSON processing logic corresponding to the JSON processing instruction is deployed in the blockchain virtual machine. An instruction set of a smart contract compiler includes the JSON processing instruction, a service smart contract compiled by using the smart contract compiler includes a contract identifier of a JSON processing smart contract, and the JSON processing smart contract is a smart contract pre-deployed in the blockchain network. The JSON processing smart contract and the service smart contract are deployed in the blockchain network.

[0337] A node in the blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0338] When executing the service initiation transaction, each node in the blockchain network invokes the service smart contract by using the blockchain virtual machine, invokes the JSON processing smart contract based on the contract identifier of the JSON processing smart contract in the service smart contract by using the blockchain virtual machine, and triggers execution of the JSON processing logic based on the JSON processing instruction in the JSON processing smart contract by using the blockchain virtual machine, to perform a JSON processing operation on data to be processed.

[0339] A blockchain system provided in the present specification includes a blockchain network.

[0340] An instruction set of a blockchain virtual machine includes an XML processing instruction, and XML processing logic corresponding to the XML processing instruction is deployed in the blockchain virtual machine. An instruction set of a smart contract compiler includes the XML processing instruction, and a service smart contract compiled by using the smart contract compiler includes the XML processing instruction. The service smart contract is deployed in the blockchain network.

[0341] A node in the blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0342] When executing the service initiation transaction, each node in the blockchain network invokes the service smart contract by using the blockchain virtual machine, and triggers execution of the XML processing logic based on the XML processing instruction in the service smart contract by using the blockchain virtual

machine, to perform an XML processing operation on data to be processed.

[0343] A blockchain system provided in the present specification includes a blockchain network.

[0344] An instruction set of a blockchain virtual machine includes an XML processing instruction, and XML processing logic corresponding to the XML processing instruction is deployed in the blockchain virtual machine. An instruction set of a smart contract compiler includes the XML processing instruction, a service smart contract compiled by using the smart contract compiler includes a contract identifier of an XML processing smart contract, and the XML processing smart contract is a smart contract pre-deployed in the blockchain network. The XML processing smart contract and the service smart contract are deployed in the blockchain network.

[0345] A node in the blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0346] When executing the service initiation transaction, each node in the blockchain network invokes the service smart contract by using the blockchain virtual machine, invokes the XML processing smart contract based on the contract identifier of the XML processing smart contract in the service smart contract by using the blockchain virtual machine, and triggers execution of the XML processing logic based on the XML processing instruction in the XML processing smart contract by using the blockchain virtual machine, to perform an XML processing operation on data to be processed.

[0347] A blockchain system provided in the present specification includes a blockchain network.

[0348] An instruction set of a blockchain virtual machine includes a transaction hash acquisition instruction, and transaction hash acquisition logic corresponding to the transaction hash acquisition instruction is deployed in the blockchain virtual machine. An instruction set of a smart contract compiler includes the transaction hash acquisition instruction, and a service smart contract compiled by using the smart contract compiler includes the transaction hash acquisition instruction. The service smart contract is deployed in the blockchain network.

[0349] A node in the blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0350] When executing the service initiation transaction, each node in the blockchain network invokes the service smart contract by using the blockchain virtual

machine, and triggers execution of the transaction hash acquisition logic based on the transaction hash acquisition instruction in the service smart contract by using the blockchain virtual machine, to obtain a transaction hash of the service initiation transaction.

[0351] A blockchain system provided in the present specification includes a blockchain network.

[0352] An instruction set of a blockchain virtual machine includes a transaction hash acquisition instruction, and transaction hash acquisition logic corresponding to the transaction hash acquisition instruction is deployed in the blockchain virtual machine. An instruction set of a smart contract compiler includes the transaction hash acquisition instruction, a service smart contract compiled by using the smart contract compiler includes a contract identifier of a transaction hash acquisition smart contract, and the transaction hash acquisition smart contract is a smart contract pre-deployed in the blockchain network. The transaction hash acquisition smart contract and the service smart contract are deployed in the blockchain network.

[0353] A node in the blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes.

[0354] When executing the service initiation transaction, each node in the blockchain network invokes the service smart contract by using the blockchain virtual machine, invokes the transaction hash acquisition smart contract based on the contract identifier of the transaction hash acquisition smart contract in the service smart contract by using the blockchain virtual machine, and triggers execution of the transaction hash acquisition logic based on the transaction hash acquisition instruction in the transaction hash acquisition smart contract by using the blockchain virtual machine, to obtain a transaction hash of the service initiation transaction.

[0355] FIG. 21 is a schematic structural diagram illustrating a blockchain system, according to some embodiments of the present specification.

[0356] The present specification further provides a computer device. The computer device includes at least a memory, a processor, and a computer program that is stored in the memory and that can run on the processor. When executing the program, the processor implements functions of the methods in the embodiments of the present specification and a method combined by at least two of the embodiments.

[0357] FIG. 22 is a more detailed schematic diagram illustrating a hardware structure of a computing device, according to some embodiments of the present

specification. The device can include a processor 2210, a memory 2220, an input/output interface 2230, a communications interface 2240, and a bus 2250. The processor 2210, the memory 2220, the input/output interface 2230, and the communications interface 2240 are communicatively connected to each other inside the device by using the bus 2250.

[0358] The processor 2210 can be implemented by using a general central processing unit (CPU), a microprocessor, an application-specific integrated circuit (ASIC), one or more integrated circuits, etc., and is configured to execute a related program, to implement the technical solutions provided in the embodiments of the present specification.

[0359] The memory 2220 can be implemented by using a read-only memory (ROM), a random access memory (RAM), a static storage device, a dynamic storage device, etc. The memory 2220 can store an operating system and another application program. When the technical solutions provided in the embodiments of the present specification are implemented by using software or firmware, related program code is stored in the memory 2220, and is invoked and executed by the processor 2210.

[0360] The input/output interface 2230 is configured to be connected to an input/output module, to input or output information. The input/output module (not shown in the figure) can be used as a component and configured in the device, or can be externally connected to the device, to provide a corresponding function. The input device can include a keyboard, a mouse device, a touchscreen, a microphone, various sensors, etc. The output device can include a monitor, a speaker, a vibrator, an indicator, etc.

[0361] The communications interface 2240 is configured to be connected to a communications module (not shown in the figure), to implement communication interaction between the device and another device. The communications module can perform communication in a wired way (for example, USB or a network cable), or can perform communication in a wireless way (for example, a mobile network, Wi-Fi, or Bluetooth).

[0362] The bus 2250 includes one channel that is used to transmit information between components (for example, the processor 2210, the memory 2220, the input/output interface 2230, and the communications interface 2240) of the device.

[0363] It is worthwhile to note that although only the processor 2210, the memory 2220, the input/output interface 2230, the communications interface 2240, and the bus

2250 of the device are shown, in a specific implementation process, the device can further include other components needed for normal running. In addition, a person skilled in the art can understand that the device can also include only components necessary for implementing the solutions in the embodiments of the present specification, but does not necessarily include all components shown in the figure.

[0364] Some embodiments of the present specification further provide a computer readable storage medium on which a computer program is stored. When executing the program, the processor implements functions of the methods in the embodiments of the present specification and a method combined by at least two of the embodiments.

[0365] The computer readable medium includes persistent, non-persistent, movable, and unmovable media that can store information by using any method or technology. The information can be a computer readable instruction, a data structure, a program module, or other data. Examples of the computer storage medium include but are not limited to a phase change random access memory (PRAM), a static random access memory (SRAM), a dynamic random access memory (DRAM), a random access memory of another type, a read-only memory (ROM), an electrically erasable programmable read-only memory (EEPROM), a flash memory or another memory technology, a compact disc read-only memory (CD-ROM), a digital versatile disc (DVD) or another optical storage, a cassette, a cassette magnetic disk storage, or another magnetic storage device or any other non-transmission medium. The computer storage medium can be configured to store information that can be accessed by a computing device. As described in the present specification, the computer readable medium does not include computer readable transitory media such as a modulated data signal and a carrier.

[0366] It can be understood from the aforementioned descriptions of the embodiments that, a person skilled in the art can clearly understand that the embodiments of the present specification can be implemented by using software and a necessary general hardware platform. Based on such an understanding, the technical solutions in the embodiments of the present specification essentially or the part contributing to the existing technology can be implemented in a form of a software product. The computer software product can be stored in a storage medium, such as a ROM/RAM, a magnetic disk, or an optical disc, and includes several instructions for instructing a computer device (which can be a personal computer, a server, a network device, etc.) to perform the method described in the embodiments of the present

specification or in some parts of the embodiments of the present specification.

[0367] The system, method, module, or unit illustrated in the aforementioned embodiments can be implemented by using a computer chip or an entity, or can be implemented by using a product having a certain function. A typical implementation device is a computer, and the computer can 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.

[0368] The embodiments in the present specification are described in a progressive way. For same or similar parts of the embodiments, references can be made to the embodiments. Each embodiment focuses on a difference from other embodiments. Particularly, an apparatus embodiment and a device embodiment are similar to a method embodiment, and therefore are described briefly. For a related part, references can be made to some descriptions in the method embodiment. The previously described apparatus embodiments are merely examples. The modules described as separate parts can or does not have to be physically separate. During implementation of the solutions in the embodiments of the present specification, functions of the modules can be implemented in one or more pieces of software and/or hardware. Some or all of the modules can be selected based on an actual need to implement the solutions in the embodiments. A person of ordinary skill in the art can understand and implement the embodiments of the present specification without creative efforts.

[0369] The aforementioned descriptions are merely specific implementations of the embodiments of the present application. It is worthwhile to note that a person of ordinary skill in the art can further make several improvements or polishing without departing from the principle of the embodiments of the present specification, and the improvements or polishing shall fall within the protection scope of the embodiments of the present specification.

CLAIMS

What is claimed is:

1. A blockchain smart contract-based transaction hash acquisition method, wherein an instruction set of a blockchain virtual machine comprises a transaction hash acquisition instruction, and transaction hash acquisition logic corresponding to the transaction hash acquisition instruction is deployed in the blockchain virtual machine;

an instruction set of a smart contract compiler comprises the transaction hash acquisition instruction, and a service smart contract compiled by using the smart contract compiler comprises the transaction hash acquisition instruction;

the service smart contract is deployed in a blockchain network; and

the transaction hash acquisition method comprises:

obtaining, by a node in the blockchain network, a service initiation transaction, and broadcasting the service initiation transaction to other nodes;

when executing the service initiation transaction, invoking, by each node in the blockchain network, the service smart contract by using the blockchain virtual machine; and

triggering, by the node, execution of the transaction hash acquisition logic based on the transaction hash acquisition instruction in the service smart contract by using the blockchain virtual machine, to obtain a transaction hash of the service initiation transaction.

2. The method according to claim 1, wherein the method further comprises:

before invoking the service smart contract, initializing, by each node in the blockchain network, a context of the blockchain virtual machine, and writing the transaction hash of the service initiation transaction into the context; and

the triggering, by the node, execution of the transaction hash acquisition logic based on the transaction hash acquisition instruction in the service smart contract by using the blockchain virtual machine, to obtain a transaction hash of the service initiation transaction specifically comprises:

triggering, by the node, execution of the transaction hash acquisition logic based on the transaction hash acquisition instruction in the service smart contract by using the blockchain virtual machine, to obtain the transaction hash of the service initiation transaction from the context.

3. A blockchain smart contract-based transaction hash acquisition method, wherein an instruction set of a blockchain virtual machine comprises a transaction hash acquisition instruction, and transaction hash acquisition logic corresponding to the transaction hash acquisition instruction is deployed in the blockchain virtual machine;

an instruction set of a smart contract compiler comprises the transaction hash acquisition instruction, a service smart contract compiled by using the smart contract compiler comprises a contract identifier of a transaction hash acquisition smart contract, and the transaction hash acquisition smart contract is a smart contract pre-deployed in a blockchain network;

the service smart contract is deployed in the blockchain network; and

the transaction hash acquisition method comprises:

obtaining, by a node in the blockchain network, a service initiation transaction, and broadcasting the service initiation transaction to other nodes;

when executing the service initiation transaction, invoking, by each node in the blockchain network, the service smart contract by using the blockchain virtual machine;

invoking, by the node, the transaction hash acquisition smart contract based on the contract identifier of the transaction hash acquisition smart contract in the service smart contract by using the blockchain virtual machine; and

triggering, by the node, execution of the transaction hash acquisition logic based on the transaction hash acquisition instruction in the transaction hash acquisition smart contract by using the blockchain virtual machine, to obtain a transaction hash of the service initiation transaction.

4. A blockchain virtual machine, configured to implement the method according to any one of claims 1 to 3, wherein

an instruction set of the blockchain virtual machine comprises a transaction hash acquisition instruction, and a service smart contract compiled by using the smart contract compiler comprises the transaction hash acquisition instruction.

5. The blockchain virtual machine according to claim 4, wherein the instruction set of the blockchain virtual machine further comprises at least one Ethereum instruction, and the Ethereum instruction is an instruction in an instruction set of an Ethereum virtual machine; and

relevant logic corresponding to each Ethereum instruction comprised in the instruction set of the blockchain virtual machine is deployed in the blockchain virtual machine.

6. A smart contract compiler, configured to implement the method according to claim 1 or 2, wherein

an instruction set of the smart contract compiler comprises a transaction hash acquisition instruction, and a service smart contract compiled by using the smart contract compiler comprises the transaction hash acquisition instruction.

7. The smart contract compiler according to claim 6, wherein the instruction set of the smart contract compiler further comprises at least one Ethereum instruction, and the Ethereum instruction is an instruction in an instruction set of an Ethereum virtual machine.

8. A smart contract compiler, configured to implement the method according to claim 3, wherein

an instruction set of the smart contract compiler comprises a transaction hash acquisition instruction, a service smart contract compiled by using the smart contract compiler comprises a contract identifier of a transaction hash acquisition smart contract, and the transaction hash acquisition smart contract is a smart contract pre-deployed in a blockchain network.

9. A blockchain system, comprising a blockchain network, wherein

an instruction set of a blockchain virtual machine comprises a transaction hash acquisition instruction, transaction hash acquisition logic corresponding to the transaction hash acquisition instruction is deployed in the blockchain virtual machine, an instruction set of a smart contract compiler comprises the transaction hash acquisition instruction, a service smart contract compiled by using the smart contract compiler comprises the transaction hash acquisition instruction, and the service smart contract is deployed in the blockchain network;

a node in the blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes; and

when executing the service initiation transaction, each node in the blockchain network invokes the service smart contract by using the blockchain virtual machine, and triggers execution of the transaction hash acquisition logic based on the transaction hash acquisition instruction in the service smart contract by using the blockchain virtual machine, to obtain a transaction hash of the service initiation transaction.

10. A blockchain system, comprising a blockchain network, wherein

an instruction set of a blockchain virtual machine comprises a transaction hash acquisition instruction, transaction hash acquisition logic corresponding to the

transaction hash acquisition instruction is deployed in the blockchain virtual machine, an instruction set of a smart contract compiler comprises the transaction hash acquisition instruction, a service smart contract compiled by using the smart contract compiler comprises a contract identifier of a transaction hash acquisition smart contract, the transaction hash acquisition smart contract is a smart contract pre-deployed in the blockchain network, and the service smart contract is deployed in the blockchain network;

a node in the blockchain network obtains a service initiation transaction, and broadcasts the service initiation transaction to other nodes; and

when executing the service initiation transaction, each node in the blockchain network invokes the service smart contract by using the blockchain virtual machine, invokes the transaction hash acquisition smart contract based on the contract identifier of the transaction hash acquisition smart contract in the service smart contract by using the blockchain virtual machine, and triggers execution of the transaction hash acquisition logic based on the transaction hash acquisition instruction in the transaction hash acquisition smart contract by using the blockchain virtual machine, to obtain a transaction hash of the service initiation transaction.

11. A computer device, comprising a memory, a processor, and a computer program that is stored in the memory and that can run on the processor, wherein when executing the program, the processor implements the method according to any one of claims 1 and 3.