

EXECUTING MULTI-PARTY TRANSACTIONS USING SMART CONTRACTS

BACKGROUND

[0001] Distributed ledger systems (DLSs), which can also be referred to as consensus networks, and/or blockchain networks, enable participating entities to securely, and immutably store data. DLSs are commonly referred to as blockchain networks without referencing any particular use case (e.g., crypto-currencies). Example types of blockchain networks can include public blockchain networks, private blockchain networks, and consortium blockchain networks. A public blockchain network is open for all entities to use the DLS, and participate in the consensus process. A private blockchain network is provided for a particular entity, which centrally controls read and write permissions. A consortium blockchain network is provided for a select group of entities, which control the consensus process, and includes an access control layer.

[0002] Smart contracts can be created between entities, and executed within a blockchain network. In some examples, a smart contract can define a transaction between the entities within the blockchain network. For example, entities in a blockchain network can call a smart contract to initiate a multi-party transaction. In some instances, each participating entity has to separately confirm the transaction before the smart contract can start executing. For example, single signatures of each participating entity.

SUMMARY

[0003] Implementations of the present specification include computer-implemented methods for verifying multi-party smart contract execution in a blockchain network. More particularly, implementations of the present specification are directed to improving efficiency and data security in smart contract execution.

[0004] In some implementations, actions include receiving first transaction information from a first node, wherein the first node is a computer node in the blockchain network, and wherein the first transaction information includes a transaction payload, a first public key, and a signed transaction payload for a transaction, verifying the signed transaction payload using the first public key, in response to the verifying the signed transaction payload, constructing an unconfirmed transaction data package, and setting a

confirmation status of the unconfirmed transaction data package, receiving second transaction information from a second node, wherein the second node is a computer node in the blockchain network, and wherein the second transaction information includes a hash of the transaction payload, a second public key, and a signed hash of the transaction payload for the transaction, verifying the second transaction information using the second public key, updating the confirmation status of the unconfirmed transaction data package, and executing the transaction payload in response to the confirmation status indicating that all parties to the transaction have confirmed the transaction. Other implementations include corresponding systems, apparatus, and computer programs, configured to perform the actions of the methods, encoded on computer storage devices.

[0005] These and other implementations may each optionally include one or more of the following features: the unconfirmed data package includes addresses of all nodes required for the execution of the multi-party transaction; the unconfirmed data package is stored in an unconfirmed transaction pool maintained by the blockchain network as a value in a key-value pair, wherein the key in the key-value pair is the hash of the transaction payload associated with the unconfirmed data package; the transaction payload includes a universally unique identifier in the blockchain network; actions further include recording execution of the transaction payload in a blockchain maintained by the blockchain network; the first and the second public keys are stored in a block of a blockchain maintained by the blockchain network; and the transaction payload includes an exchange of at least one asset between the first node and the second node.

[0006] The present specification also provides one or more non-transitory computer-readable storage media coupled to one or more processors and having instructions stored thereon which, when executed by the one or more processors, cause the one or more processors to perform operations in accordance with implementations of the methods provided herein.

[0007] The present specification further provides a system for implementing the methods provided herein. The system includes one or more processors, and a computer-readable storage medium coupled to the one or more processors having instructions stored thereon which, when executed by the one or more processors, cause the one or

more processors to perform operations in accordance with implementations of the methods provided herein.

[0008] It is appreciated that methods in accordance with the present specification may include any combination of the aspects and features described herein. That is, methods in accordance with the present specification are not limited to the combinations of aspects and features specifically described herein, but also include any combination of the aspects and features provided.

[0009] The details of one or more implementations of the present specification are set forth in the accompanying drawings and the description below. Other features and advantages of the present specification will be apparent from the description and drawings, and from the claims.

DESCRIPTION OF DRAWINGS

[0010] FIG. 1 depicts an example environment that can be used to execute implementations of the present specification.

[0011] FIG. 2 depicts an example conceptual architecture in accordance with implementations of the present specification.

[0012] FIG. 3 depicts an example signal diagram for executing a multi-party transaction in accordance with implementations of the present specification.

[0013] FIG. 4 depicts an example process that can be executed in accordance with implementations of the present specification.

[0014] Like reference symbols in the various drawings indicate like elements.

DETAILED DESCRIPTION

[0015] Implementations of the present specification include computer-implemented methods for verifying multi-party smart contract execution in a blockchain network. More particularly, implementations of the present specification are directed to maintaining a transaction confirmation status of a multi-party transaction using a smart contract, and executing the transaction after confirmation of all parties has been received. In some implementations, actions include receiving first transaction information from a first node, wherein the first node is a computer node in the blockchain network, and

wherein the first transaction information includes a transaction payload, a first public key, and a signed transaction payload for a transaction, verifying the signed transaction payload using the first public key, in response to the verifying the signed transaction payload, constructing an unconfirmed transaction data package, and setting a confirmation status of the unconfirmed transaction data package, receiving second transaction information from a second node, wherein the second node is a computer node in the blockchain network, and wherein the second transaction information includes a hash of the transaction payload, a second public key, and a signed hash of the transaction payload for the transaction, verifying the second transaction information using the second public key, updating the confirmation status of the unconfirmed transaction data package, and executing the transaction payload in response to the confirmation status indicating that all parties to the transaction have confirmed the transaction.

[0016] To provide further context for implementations of the present specification, and as introduced above, distributed ledger systems (DLSs), which can also be referred to as consensus networks (e.g., made up of peer-to-peer nodes), and blockchain networks, enable participating entities to securely, and immutably conduct transactions, and store data. Although the term blockchain is generally associated with the Bitcoin cryptocurrency network, blockchain is used herein to generally refer to a DLS without reference to any particular use case. As introduced above, a blockchain network can be provided as a public blockchain network, a private blockchain network, or a consortium blockchain network. Implementations of the present specification are described in further detail herein with reference to a public blockchain network, which is public among the participating entities. It is contemplated, however, that implementations of the present specification can be realized in any appropriate type of blockchain network.

[0017] To provide further context for implementations of the present specification, in blockchain networks, applications can be developed, tested, and deployed for execution within the blockchain network. An example application can include, without limitation, a smart contract. A smart contract can be described as digital representations of real-world, legal contracts having contractual terms affecting various parties. A smart contract is implemented, stored, updated (as needed), and executed within, in the example context, a

consortium blockchain network. Contract parties associated with the smart contract (e.g., buyers and sellers) are represented as nodes in the consortium blockchain network.

[0018] In some examples, a smart contract can store data, which can be used to record information, facts, associations, balances and any other information needed to implement logic for contract execution. Smart contracts can be described as a computer-executable program consisting of functions, where an instance of the smart contract can be created, and functions invoked for execution of the logic therein.

[0019] In technical terms, smart contracts can be implemented based on objects and object-oriented classes. For example, terms and components of the smart contract can be represented as objects that are handled by applications implementing the smart contracts. A smart contract (or an object in the smart contract) can call another smart contract (or an object in the same smart contract) just like other object-oriented objects. Calls that are made by an object can be, for example, a call to create, update, delete, propagate, or communicate with objects of another class. Calls between objects can be implemented as functions, methods, application programming interfaces (APIs), or other calling mechanisms. For example, a first object can call a function to create a second object.

[0020] Implementations of the present specification are described in further detail herein in view of the above context. More particularly, and as introduced above, implementations of the present specification are directed to maintaining a transaction confirmation status of a multi-party transaction using a smart contract, and executing the transaction after confirmation of all parties has been received.

[0021] FIG. 1 depicts an example environment 100 that can be used to execute implementations of the present specification. In some examples, the example environment 100 enables entities to participate in a blockchain network 102. The blockchain network 102 can be a public blockchain network, a private blockchain network, or a consortium blockchain network. The example environment 100 includes computing devices 106, 108, and a network 110. In some examples, the network 110 includes a local area network (LAN), wide area network (WAN), the Internet, or a combination thereof, and connects websites, user devices (e.g., computing devices), and back-end systems. In some examples, the network 110 can be accessed over a wired and/or a wireless communications link.

[0022] In the depicted example, the computing systems 106, 108 can each include any appropriate computing system that enables participation as a node in the blockchain network 102. Example computing devices include, without limitation, a server, a desktop computer, a laptop computer, a tablet computing device, and a smartphone. In some examples, the computing systems 106, 108 each hosts one or more computer-implemented services for interacting with the blockchain network 102. For example, the computing system 106 can host computer-implemented services of a first entity (e.g., user A), such as transaction management system that the first entity uses to manage its transactions with one or more other entities (e.g., other users). The computing system 108 can host computer-implemented services of a second entity (e.g., user B), such as transaction management system that the second entity uses to manage its transactions with one or more other entities (e.g., other users). In the example of FIG. 1, the blockchain network 102 is represented as a peer-to-peer network of nodes, and the computing systems 106, 108 provide nodes of the first entity, and second entity respectively, which participate in the blockchain network 102.

[0023] FIG. 2 depicts an example conceptual architecture 200 in accordance with implementations of the present specification. The example conceptual architecture 200 includes an entity layer 202, a hosted services layer 204, and a blockchain network layer 206. In the depicted example, the entity layer 202 includes three entities, Entity_1 (E1), Entity_2 (E2), and Entity_3 (E3), each entity having a respective transaction management system 208.

[0024] In the depicted example, the hosted services layer 204 includes interfaces 210 for each transaction management system 210. In some examples, a respective transaction management system 208 communicates with a respective interface 210 over a network (e.g., the network 110 of FIG. 1) using a protocol (e.g., hypertext transfer protocol secure (HTTPS)). In some examples, each interface 210 provides a communication connection between a respective transaction management system 208, and the blockchain network layer 206. More particularly, the interface 210 communicates with a blockchain network 212 of the blockchain network layer 206. In some examples, communication between an interface 210, and the blockchain network layer 206 is conducted using remote procedure calls (RPCs). In some examples, the interfaces 210 “host” blockchain network nodes for

the respective transaction management systems 208. For example, the interfaces 210 provide the application programming interface (API) for access to blockchain network 212.

[0025] As described herein, the blockchain network 212 is provided as a peer-to-peer network including a plurality of nodes 214 that immutably record information in a blockchain 216. Although a single blockchain 216 is schematically depicted, multiple copies of the blockchain 216 are provided, and are maintained across the blockchain network 212. For example, each node 214 stores a copy of the blockchain. In some implementations, the blockchain 216 stores information associated with transactions that are performed between two or more entities participating in the blockchain network.

[0026] As described in further detail herein, implementations of the present specification are directed to execution of multi-party transactions within blockchain networks. In accordance with implementations of the present specification, a smart contract executes in the blockchain network, and verifies signatures of users (parties) participating in a transaction. In some implementations, the smart contract includes an unconfirmed data structure, in which a transaction status is maintained. Upon confirmation of all parties to the transaction, the transaction is executed.

[0027] FIG. 3 depicts an example signal diagram 300 for executing a multi-party transaction in accordance with implementations of the present specification. The example signal diagram 300 of FIG. 3 includes a user A 302 (e.g., a node in a blockchain network), a user B 304 (e.g., a node in the blockchain network), a smart contract 306 executing within the blockchain network, and a contract manager 308.

[0028] The user A 302 initiates a transaction in the blockchain network by constructing a transaction payload (310). A transaction payload is a data package that provides the details of the intended transaction. For example, the user A 302 can include in the payload the address of all the participating entities (e.g., the user A 302 and the user B 304) within the blockchain network, an asset, and/or value that is the subject of the transaction, and the like.

[0029] The user A 302 digitally signs the transaction payload (312). In some implementations, the user A 302 uses asymmetric cryptography technology to sign the transaction payload. For example, the user A 302 can have a key pair associated

therewith, the key pair including a public key (e.g., pubkey_A that can be known to anyone participating in the blockchain network), and a private key (e.g., privkey_A that is only known to the user A). The user A302 signs the transaction payload with the private key to provide a hash value (e.g., represented as sig_A(payload)). The following example transaction information package can be provided: [payload, pubkey_A, sig_A(payload)].

[0030] In accordance with implementations of the present specification, the user A 302 submits (314) the transaction information package, which includes the transaction payload, the digitally signed transaction payload, and the public key to the smart contract 306. The digital signature of the user A 302 is verified using the public key (316). In some examples, the smart contract 306 verifies that the transaction is valid (e.g., the transaction was sent from the user A 302), the smart contract 306 begins to execute the transaction payload (314). In some examples, the blockchain network verifies the digital signature of the user A 302 using the public key.

[0031] If the transaction is a multi-party transaction, that is, a transaction involving at least two participating entities, the smart contract 306 constructs an unconfirmed transaction data package using the transaction payload, stored the unconfirmed transaction data package in an unconfirmed transaction pool, and sets a confirmation (318). An example confirmation status of the unconfirmed transaction data package can include [A: confirmed, B: unconfirmed].

[0032] In some examples, the unconfirmed transaction pool can be a data stored (e.g., an associative array, a table) that includes key-value pairs maintained by the smart contract 306. The key in the unconfirmed transaction pool is a hash value of a transaction payload, and the value in the unconfirmed transaction pool is the corresponding unconfirmed transaction data package. For example, when the user A 302 submits the transaction payload to the smart contract 306, the corresponding entry in the unconfirmed transaction pool can be represented as: (hash(payload), [payload, node_A_address, node_B_address, node_A_confirmation_status, node_B_confirmation_status]).

[0033] In addition to submitting the signed transaction payload to the smart contract 306, the user A 302 also submits (320) the signed transaction payload to other participating entities (e.g., the user B 304). The user B 304 uses the public key of the user

A 302 to verify the signed payload, hashes the payload, and signs the hashed payload with the private key of the user B 304 (322). The user B 304 submits the hashed payload, the signed hashed payload, and its public key to the smart contract 306.

[0034] The smart contract 306 verifies (326) the digital signature of the user B 304 using the public key of the user B 304. The smart contract 306 uses the hashed payload as a key to locate the corresponding unconfirmed transaction data package within the unconfirmed transaction pool (328). The hash function used by the user B 304 is the same hash function used by the smart contract 306 when constructing the unconfirmed transaction data package. The smart contract 306 updates the unconfirmed transaction data package by changing the confirmation status of the user B 302 to confirmed (328) (e.g., [A: confirmed, B: confirmed]).

[0035] In an alternative implementation, the unconfirmed transaction data package can be located by assigning each transaction payload a universally unique identifier (UUID). Instead of signing the hashed payload, the user B 304 signs the entire payload similar to that performed by the user A 302. The smart contract 306 uses the UUID to locate the unconfirmed transaction pending confirmation of the user B 304.

[0036] After all parties (e.g., the user A 302 and the user B 304) have confirmed the transaction, the smart contract 306 executes the transaction (330). If the transaction involves more than two entities, each of the entities other than the initiating entity has to separately hash and sign the transaction payload. In some examples, execution of the transaction includes submitting the transaction to the blockchain network for consensus processing, and packaging of the transaction within a block that is added to the blockchain.

[0037] After the transaction concludes (e.g., consensus processing is successful, and the transaction is added to the blockchain), the smart contract 306 removes the transaction from the unconfirmed transaction pool (332).

[0038] In some implementations, the contract manager 308 periodically checks the unconfirmed transaction pool for an expiration condition. In some examples, an unconfirmed transaction data package only stays in the unconfirmed transaction pool for a predetermined period of time. If the predetermined period of time expires (e.g., all parties do not confirm the transaction within the predetermined period of time, the

unconfirmed transaction is deleted (334). Imposing this time limit ensures that unwanted transactions submitted by malicious entities do not occupy resources of the blockchain network.

[0039] FIG. 4 depicts an example process 400 that can be executed in accordance with implementations of the present specification. In some examples, the example process 400 is provided using one or more computer-executable programs executed by one or more computing devices. For example, at least a portion of the example process 400 can be executed by a smart contract executing within a blockchain network (e.g., the smart contract 306 of FIG. 3 executing within the blockchain network 212 of FIG. 2).

[0040] A signed transaction is received (402). For example, the smart contract 306 receives a transaction from the user A 302 (e.g., the user A 302 sends a signed transaction package to the smart contract 306). It is determined whether a signature of the signed transaction is valid (404). For example, the smart contract 306 uses the public key of the user A 302 to determine whether the signature of the transaction is valid. If the signature is not valid, an error is indicated, and the example process 400 ends.

[0041] If the signature is valid, an unconfirmed transaction package is provided and is stored in an unconfirmed transaction pool (408). For example, and as described herein, the smart contract 306 provides a key for the transaction (e.g., based on a hash, based on a UUID), and stores the transaction in the unconfirmed transaction pool with the key. A party status is set (410). For example, the smart contract 306 sets a party status of the transaction to [A: confirmed, B: unconfirmed].

[0042] It is determined whether all parties to the transaction have confirmed the transaction (412). If all parties have confirmed the transaction, the transaction is executed. For example, the smart contract 306 submits the transaction to the blockchain network for consensus processing. In some examples, the transaction is deleted from the unconfirmed transaction pool.

[0043] It is determined whether the transaction has expired (416). For example, the smart contract 306 receives a periodic signal from the contract manager 308, which triggers the smart contract 306 to determine whether the transaction has expired (e.g., has been unconfirmed for greater than or equal to a predetermined period of time). If the transaction has expired, the transaction is deleted from the unconfirmed transaction pool.

(418). If the transaction has not expired, it is determined whether another transaction has been received (420). If another transaction has not been received, the example process 400 loops back to check expiration.

[0044] If another transaction has been received, it is determined whether a signature of the transaction is valid (422). For example, the smart contract 306 receives a transaction from the user B 304 (e.g., the user B 304 sends a signed transaction package to the smart contract 306). If the signature is not valid, an error is indicated 424, and the example process 400 loops back. If the signature is valid, it is determined whether the transaction corresponds to a transaction stored in the unconfirmed transaction pool (426). For example, the smart contract 306 uses a value of the received transaction (e.g., hash, UUID) to search for a corresponding key in the unconfirmed transaction pool. If the transaction is not in the unconfirmed transaction pool, the transaction can be considered a new transaction, and the example process 400 loops back to add the transaction to the unconfirmed transaction pool. If the transaction is in the unconfirmed transaction pool, the example process 400 loops back to update the status of the parties (410) (e.g., the smart contract 306 sets a party status of the transaction to [A: confirmed, B: confirmed]), and determine whether all parties have confirmed that transaction (412), as described herein.

[0045] The features described may be implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations of them. The apparatus may be implemented in a computer program product tangibly embodied in an information carrier (e.g., in a machine-readable storage device) for execution by a programmable processor; and method steps may be performed by a programmable processor executing a program of instructions to perform functions of the described implementations by operating on input data and generating output. The described features may be implemented advantageously in one or more computer programs that are executable on a programmable system including at least one programmable processor coupled to receive data and instructions from, and to transmit data and instructions to, a data storage system, at least one input device, and at least one output device. A computer program is a set of instructions that may be used, directly or indirectly, in a computer to perform a certain activity or bring about a certain result. A computer program may be written in any form

of programming language, including compiled or interpreted languages, and it may be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or another unit suitable for use in a computing environment.

[0046] Suitable processors for the execution of a program of instructions include, by way of example, both general and special purpose microprocessors, and the sole processor or one of the multiple processors of any kind of computer. Generally, a processor will receive instructions and data from a read-only memory or a random access memory or both. Elements of a computer may include a processor for executing instructions and one or more memories for storing instructions and data. Generally, a computer may also include, or be operatively coupled to communicate with, one or more mass storage devices for storing data files; such devices include magnetic disks, such as internal hard disks and removable disks; magneto-optical disks; and optical disks. Storage devices suitable for tangibly embodying computer program instructions and data include all forms of non-volatile memory, including by ways of example semiconductor memory devices, such as EPROM, EEPROM, and flash memory devices; magnetic disks such as internal hard disks and removable disks; magneto-optical disks; and CD-ROM and DVD-ROM disks. The processor and the memory may be supplemented by, or incorporated in, application-specific integrated circuits (ASICs).

[0047] To provide for interaction with a user, the features may be implemented on a computer having a display device such as a cathode ray tube (CRT) or liquid crystal display (LCD) monitor for displaying information to the user and a keyboard and a pointing device such as a mouse or a trackball by which the user may provide input to the computer.

[0048] The features may be implemented in a computer system that includes a back-end component, such as a data server, or that includes a middleware component, such as an application server or an Internet server, or that includes a front-end component, such as a client computer having a graphical user interface or an Internet browser, or any combination of them. The components of the system may be connected by any form or medium of digital data communication such as a communication network. Examples of communication networks include, e.g., a local area network (LAN), a wide area network (WAN), and the computers and networks forming the Internet.

[0049] The computer system may include clients and servers. A client and server are generally remote from each other and typically interact through a network, such as the described one. The relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other.

[0050] In addition, the logic flows depicted in the figures do not require the particular order shown, or sequential order, to achieve desirable results. In addition, other steps may be provided, or steps may be eliminated, from the described flows, and other components may be added to, or removed from, the described systems. Accordingly, other implementations are within the scope of the following claims.

[0051] A number of implementations of the present specification have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the present specification. Accordingly, other implementations are within the scope of the following claims.

WHAT IS CLAIMED IS:

1. A computer-implemented method for executing a multi-party transaction in a blockchain network, the method comprising:

- receiving first transaction information for the multi-party transaction from a first computer node in the blockchain network, wherein the first transaction information comprises a transaction payload, a first public key, and a first signature associated with the first computer node;

- verifying the first signature;

- in response to the verifying the first signature:

- constructing an unconfirmed transaction data package;

- storing the unconfirmed transaction data package as a value in a key-value pair, wherein a key in the key-value pair is a hash of the transaction payload; and

- setting a confirmation status of the unconfirmed transaction data package;

- receiving second transaction information from a second computer node in the blockchain network, wherein the second transaction information comprises the hash of the transaction payload, a second public key, and a second signature associated with the second computer node;

- verifying the second signature;

- in response to verifying the second signature, locating the unconfirmed transaction data package using the hash of the transaction payload comprised in the second transaction information;

- updating the confirmation status of the unconfirmed transaction data package;
- and

- executing the transaction payload in response to the confirmation status indicating that all parties to the multi-party transaction have confirmed the multi-party transaction.

2. The computer-implemented method of claim 1, wherein the unconfirmed transaction data package comprises addresses of all computer nodes required for execution of the transaction payload.

3. The computer-implemented method of claim 1, wherein the unconfirmed transaction data package is stored in an unconfirmed transaction pool maintained by the blockchain network.
4. The computer-implemented method of claim 1, wherein the transaction payload includes a universally unique identifier in the blockchain network.
5. The computer-implemented method of claim 1, further comprising recording execution of the transaction payload in a blockchain maintained by the blockchain network.
6. The computer-implemented method of claim 1, wherein the first public key and the second public key are stored in a block of a blockchain maintained by the blockchain network.
7. The computer-implemented method of claim 1, wherein the transaction payload comprises an exchange of at least one asset between the first computer node and the second computer node.
8. One or more computer-readable storage media encoded with instructions that, when executed by one or more computers, cause the one or more computers to perform operations in accordance with the computer-implemented method of one or more of claims 1-7.
9. A system, comprising:
 - one or more computers; and
 - one or more computer-readable memories coupled to the one or more computers and configured with instructions executable by the one or more computers to perform operations in accordance with the computer-implemented method of one or more of claims 1-7.