

## **SECURELY EXECUTING SMART CONTRACT OPERATIONS IN A TRUSTED EXECUTION ENVIRONMENT**

### **TECHNICAL FIELD**

[01] This specification relates to securely executing smart contract operations in a trusted execution environment.

### **BACKGROUND**

[02] 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. An example of a type of blockchain network can include consortium blockchain networks provided for a select group of entities, which control the consensus process, and includes an access control layer.

[03] Smart contracts are programs that execute on blockchains. A smart contract contains a set of pre-defined rules under which the parties to that smart contract agree to interact with each other. If the pre-defined rules of the smart contract are met, the agreement defined in the smart contract is automatically enforced. A smart contract is usually tamper resistant and facilitates, verifies, and enforces the negotiation or performance of an agreement or transaction.

[04] In a consortium blockchain network, because only a selected group of nodes control the consensus process, an attacker must gain control over a relatively small number of nodes in order to influence the consensus process. Although techniques have been proposed for addressing these types of security issues in consortium blockchain networks, a more effective and secure solution would be advantageous.

### **SUMMARY**

[05] This specification describes technologies for securely executing requested smart contract operations in a trusted execution environments (TEE) executed by a blockchain node. More particularly, embodiments of this specification enable the blockchain node to perform smart contract operations in a secure and verifiable manner within a TEE such that parties

can trust that the environment in which the operations are executed has not been tampered with or compromised.

[06] This 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 embodiments of the methods provided herein.

[07] This 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 embodiments of the methods provided herein.

[08] It is appreciated that methods in accordance with this specification may include any combination of the aspects and features described herein. That is, methods in accordance with this 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.

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

#### **BRIEF DESCRIPTION OF THE DRAWINGS**

[010] FIG. 1 is a diagram illustrating an example of an environment that can be used to execute embodiments of this specification.

[011] FIG. 2 is a diagram illustrating an example of an architecture in accordance with embodiments of this specification.

[012] FIG. 3 is a diagram illustrating an example of a system in accordance with embodiments of this specification.

[013] FIG. 4 is a diagram illustrating an example of a system in accordance with embodiments of this specification.

[014] FIG. 5 depicts an example of a process that can be executed in accordance with embodiments of this specification.

[015] FIG. 6 depicts examples of modules of an apparatus in accordance with embodiments of this specification.

[016] Like reference numbers and designations in the various drawings indicate like elements.

### DETAILED DESCRIPTION

[017] This specification describes technologies for securely executing requested smart contract operations in a trusted execution environments (TEE) executed by a blockchain node. More particularly, embodiments of this specification enable the blockchain node to perform smart contract operations in a secure and verifiable manner within a TEE such that parties can trust that the environment in which the operations are executed has not been tampered with or compromised.

[018] To provide further context for embodiments of this 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 particular networks, and/or use cases, blockchain is used herein to generally refer to a DLS without reference to any particular use case.

[019] A blockchain is a data structure that stores transactions in a way that the transactions are immutable. Thus, transactions recorded on a blockchain are reliable and trustworthy. A blockchain includes one or more blocks. Each block in the chain is linked to a previous block immediately before it in the chain by including a cryptographic hash of the previous block. Each block also includes a timestamp, its own cryptographic hash, and one or more transactions. The transactions, which have already been verified by the nodes of the blockchain network, are hashed and encoded into a Merkle tree. A Merkle tree is a data structure in which data at the leaf nodes of the tree is hashed, and all hashes in each branch of the tree are concatenated at the root of the branch. This process continues up the tree to the

root of the entire tree, which stores a hash that is representative of all data in the tree. A hash purporting to be of a transaction stored in the tree can be quickly verified by determining whether it is consistent with the structure of the tree.

[020] Whereas a blockchain is a decentralized or at least partially decentralized data structure for storing transactions, a blockchain network is a network of computing nodes that manage, update, and maintain one or more blockchains by broadcasting, verifying and validating transactions, etc. As introduced above, a blockchain network can be provided as a public blockchain network, a private blockchain network, or a consortium blockchain network. Embodiments of this specification are described in further detail herein with reference to a consortium blockchain network. It is contemplated, however, that embodiments of this specification can be realized in any appropriate type of blockchain network.

[021] In general, a consortium blockchain network is private among the participating entities. In a consortium blockchain network, the consensus process is controlled by an authorized set of nodes, which can be referred to as consensus nodes, one or more consensus nodes being operated by a respective entity (e.g., a financial institution, insurance company). For example, a consortium of ten (10) entities (e.g., financial institutions, insurance companies) can operate a consortium blockchain network, each of which operates at least one node in the consortium blockchain network.

[022] In some examples, within a consortium blockchain network, a global blockchain is provided as a blockchain that is replicated across all nodes. That is, all consensus nodes are in perfect state consensus with respect to the global blockchain. To achieve consensus (e.g., agreement to the addition of a block to a blockchain), a consensus protocol is implemented within the consortium blockchain network. For example, the consortium blockchain network can implement a practical Byzantine fault tolerance (PBFT) consensus, described in further detail below.

[023] FIG. 1 is a diagram illustrating an example of an environment 100 that can be used to execute embodiments of this specification. In some examples, the example environment 100 enables entities to participate in a consortium blockchain network 102. 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 web sites, 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.

[024] In the depicted example, the computing systems 106, 108 can each include any appropriate computing system that enables participation as a node in the consortium 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 hosts one or more computer-implemented services for interacting with the consortium blockchain network 102. For example, the computing system 106 can host computer-implemented services of a first entity (e.g., user A), such as a 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 a 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 consortium 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 consortium blockchain network 102.

[025] FIG. 2 depicts an example of a conceptual architecture 200 in accordance with embodiments of this specification. The 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 participants, Participant A, Participant B, and Participant C, each participant having a respective transaction management system 208.

[026] 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 communication connection between a respective transaction management system 208, and the blockchain network layer

206. More particularly, the interface 210 communicate 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.

[027] 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 embodiments, the blockchain 216 stores information associated with transactions that are performed between two or more entities participating in the consortium blockchain network.

[028] A blockchain (e.g., the blockchain 216 of FIG. 2) is made up of a chain of blocks, each block storing data. Example data includes transaction data representative of a transaction between two or more participants. While transactions are used herein by way of non-limiting example, it is contemplated that any appropriate data can be stored in a blockchain (e.g., documents, images, videos, audio). Example transactions can include, without limitation, exchanges of something of value (e.g., assets, products, services, currency). The transaction data is immutably stored within the blockchain. That is, the transaction data cannot be changed.

[029] Before storing in a block, the transaction data is hashed. Hashing is a process of transforming the transaction data (provided as string data) into a fixed-length hash value (also provided as string data). It is not possible to un-hash the hash value to obtain the transaction data. Hashing ensures that even a slight change in the transaction data results in a completely different hash value. Further, and as noted above, the hash value is of fixed length. That is, no matter the size of the transaction data the length of the hash value is fixed. Hashing includes processing the transaction data through a hash function to generate the hash value. An example of a hash function includes, without limitation, the secure hash algorithm (SHA)-256, which outputs 256-bit hash values.

**[030]** Transaction data of multiple transactions are hashed and stored in a block. For example, hash values of two transactions are provided, and are themselves hashed to provide another hash. This process is repeated until, for all transactions to be stored in a block, a single hash value is provided. This hash value is referred to as a Merkle root hash, and is stored in a header of the block. A change in any of the transactions will result in change in its hash value, and ultimately, a change in the Merkle root hash.

**[031]** Blocks are added to the blockchain through a consensus protocol. Multiple nodes within the blockchain network participate in the consensus protocol, and perform work to have a block added to the blockchain. Such nodes are referred to as consensus nodes. PBFT, introduced above, is used as a non-limiting example of a consensus protocol. The consensus nodes execute the consensus protocol to add transactions to the blockchain, and update the overall state of the blockchain network.

**[032]** In further detail, the consensus node generates a block header, hashes all of the transactions in the block, and combines the hash value in pairs to generate further hash values until a single hash value is provided for all transactions in the block (the Merkle root hash). This hash is added to the block header. The consensus node also determines the hash value of the most recent block in the blockchain (i.e., the last block added to the blockchain). The consensus node also adds a nonce value, and a timestamp to the block header.

**[033]** In general, PBFT provides a practical Byzantine state machine replication that tolerates Byzantine faults (e.g., malfunctioning nodes, malicious nodes). This is achieved in PBFT by assuming that faults will occur (e.g., assuming the existence of independent node failures, and/or manipulated messages sent by consensus nodes). In PBFT, the consensus nodes are provided in a sequence that includes a primary consensus node, and backup consensus nodes. The primary consensus node is periodically changed. Transactions are added to the blockchain by consensus nodes within the blockchain network reaching an agreement as to the world state of the blockchain network. In this process, messages are transmitted between consensus nodes, and each consensus nodes proves that a message is received from a specified peer node, and verifies that the message was not modified during transmission.

[034] In PBFT, the consensus protocol is provided in multiple phases with all consensus nodes beginning in the same state. To begin, a client sends a request to the primary consensus node to invoke a service operation (e.g., execute a transaction within the blockchain network). In response to receiving the request, the primary consensus node multicasts the request to the backup consensus nodes. The backup consensus nodes execute the request, and each sends a reply to the client. The client waits until a threshold number of replies are received. In some examples, the client waits for  $f+1$  replies to be received, where  $f$  is the maximum number of faulty consensus nodes that can be tolerated within the blockchain network. The final result is that a sufficient number of consensus nodes come to an agreement on the order of the record that is to be added to the blockchain, and the record is either accepted, or rejected.

[035] In some blockchain networks, cryptography is implemented to maintain privacy of transactions. For example, if two nodes want to keep a transaction private, such that other nodes in the blockchain network cannot discern details of the transaction, the nodes can encrypt the transaction data. Example cryptography includes, without limitation, symmetric encryption, and asymmetric encryption. Symmetric encryption refers to an encryption process that uses a single key for both encryption (generating ciphertext from plaintext), and decryption (generating plaintext from ciphertext). In symmetric encryption, the same key is available to multiple nodes, so each node can en-/de-crypt transaction data.

[036] Asymmetric encryption uses keys pairs that each include a private key, and a public key, the private key being known only to a respective node, and the public key being known to any or all other nodes in the blockchain network. A node can use the public key of another node to encrypt data, and the encrypted data can be decrypted using other node's private key. For example, and referring again to FIG. 2, Participant A can use Participant B's public key to encrypt data, and send the encrypted data to Participant B. Participant B can use its private key to decrypt the encrypted data (ciphertext) and extract the original data (plaintext). Messages encrypted with a node's public key can only be decrypted using the node's private key.

[037] Asymmetric encryption is used to provide digital signatures, which enables participants in a transaction to confirm other participants in the transaction, as well as the validity of the transaction. For example, a node can digitally sign a message, and another

node can confirm that the message was sent by the node based on the digital signature of Participant A. Digital signatures can also be used to ensure that messages are not tampered with in transit. For example, and again referencing FIG. 2, Participant A is to send a message to Participant B. Participant A generates a hash of the message, and then, using its private key, encrypts the hash to provide a digital signature as the encrypted hash. Participant A appends the digital signature to the message, and sends the message with digital signature to Participant B. Participant B decrypts the digital signature using the public key of Participant A, and extracts the hash. Participant B hashes the message and compares the hashes. If the hashes are same, Participant B can confirm that the message was indeed from Participant A, and was not tampered with.

**[038]** In some embodiments, nodes of the blockchain network, and/or nodes that communicate with the blockchain network can operate using TEEs. At a high-level, a TEE is a trusted environment within hardware (one or more processors, memory) that is isolated from the hardware's operating environment (e.g., operating system (OS), basic input/output system (BIOS)). In further detail, a TEE is a separate, secure area of a processor that ensures the confidentiality, and integrity of code executing, and data loaded within the main processor. Within a processor, the TEE runs in parallel with the OS. At least portions of so-called trusted applications (TAs) execute within the TEE, and have access to the processor and memory. Through the TEE, the TAs are protected from other applications running in the main OS. Further, the TEE cryptographically isolates TAs from one another inside the TEE.

**[039]** An example of a TEE includes Software Guard Extensions (SGX) provided by Intel Corporation of Santa Clara, California, United States. Although SGX is discussed herein by way of example, it is contemplated that embodiments of this specification can be realized using any appropriate TEE.

**[040]** SGX provides a hardware-based TEE. In SGX, the trusted hardware is the die of the central processing unit (CPU), and a portion of physical memory is isolated to protect select code and data. The isolated portions of memory are referred to as enclaves. More particularly, an enclave is provided as an enclave page cache (EPC) in memory and is mapped to an application address space. The memory (e.g., DRAM) includes a preserved random memory (PRM) for SGX. The PRM is a continuous memory space in the lowest

BIOS level and cannot be accessed by any software. Each EPC is a memory set (e.g., 4 KB) that is allocated by an OS to load application data and code in the PRM. EPC metadata (EPCM) is the entry address for respective EPCs and ensures that each EPC can only be shared by one enclave. That is, a single enclave can use multiple EPCs, while an EPC is dedicated to a single enclave.

[041] During execution of a TA, the processor operates in a so-called enclave mode when accessing data stored in an enclave. Operation in the enclave mode enforces an extra hardware check to each memory access. In SGX, a TA is compiled to a trusted portion, and an untrusted portion. The trusted portion is inaccessible by, for example, OS, BIOS, privileged system code, virtual machine manager (VMM), system management mode (SMM), and the like. In operation, the TA runs and creates an enclave within the PRM of the memory. A trusted function executed by the trusted portion within the enclave is called by the untrusted portion, and code executing within the enclave sees the data as plaintext data (unencrypted), and external access to the data is denied. The trusted portion provides an encrypted response to the call, and the TA continues to execute.

[042] An attestation process can be performed to verify that expected code (e.g., the trusted portion of the TA) is securely executing within the SGX-provided TEE. In general, the attestation process includes a TA receiving an attestation request from a challenger (e.g., another node in the blockchain network, a key management system (KMS) of the blockchain network). In response, the TA requests that its enclave produce a remote-attestation, also referred to as a quote. Producing the remote-attestation includes a local-attestation being sent from the enclave to a so-called quoting enclave, which verifies the local-attestation, and converts the local-attestation into the remote-attestation by signing the local-attestation using an asymmetric attestation key. The remote-attestation (quote) is provided to the challenger (e.g., KMS of the blockchain network).

[043] The challenger uses an attestation verification service to verify the remote-attestation. For SGX, Intel provides the Intel Attestation Service (IAS), which receives the remote-attestation from the challenger, and verifies the remote-attestation. More particularly, the IAS processes the remote-attestation, and provides a report (e.g., attestation verification report (AVR)), which indicates whether the remote-attestation is verified. If not verified, an

error can be indicated. If verified (the expected code is securely executing in the TEE), the challenger can start, or continue interactions with the TA. For example, in response to the verification, the KMS (as challenger) can issue asymmetric encryption keys (e.g., a public-key and private-key pair) to the node executing the TEE (e.g., through a key exchange process, such as elliptical curve Diffie-Hellman (ECDH)) to enable the node to securely communicate with other nodes, and/or clients.

**[044]** In some blockchain networks, so-called smart contracts can be executed. Smart contracts 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. In some examples, the contract parties can include entities (e.g., business enterprises) that are associated with the smart contract (e.g., as parties to the smart contract).

**[045]** In further detail, smart contracts are provided as computer-executable programs that execute on blockchains (e.g., a node within a blockchain network). A smart contract contains a set of pre-defined rules under which the parties to that smart contract agree to interact with each other. If the pre-defined rules of the smart contract are met, the agreement defined in the smart contract is automatically enforced. A smart contract is usually tamper resistant and facilitates, verifies, and enforces the negotiation or performance of an agreement or transaction.

**[046]** FIG. 3 is a diagram illustrating an example of a system 300 in accordance with embodiments of this specification. As shown, system 300 includes a blockchain network 302 including blockchain nodes 304a-d. The blockchain nodes 304a-d include service TEEs 306a-d and key management (KM) TEEs 308a-d. The nodes 304a-d have access to smart contract service logic 330. A key management center 310 is communicably coupled to the nodes 304a-d.

**[047]** Each of the nodes 304a-d is blockchain node participating in the blockchain network 302 and contributing to the maintenance of a blockchain associated with the blockchain network 302 (not shown). As described above, the nodes 304a-d can participate

in a consensus process associated with the blockchain network 302, can collect transactions into blocks for addition to the blockchain, can process transactions requested by users of the blockchain network 302, can execute operations encoded in smart contracts, and perform other tasks related to the management of the blockchain. In some embodiments, each node can be a computing device (e.g., a server) including one or more processors, storage devices, and other components. In some cases, the nodes 304a-d communicate over a communications network (not shown) to each other and to other nodes participating in the blockchain network 302. For the remainder of the description of FIG. 3, node 304a will be described as an example, with the understanding that nodes 304b-d can also include the features of node 304a.

**[048]** Node 304a includes a service TEE 306a. In some embodiments, the service TEE 306a is a secure application environment implemented using a TEE technology (e.g., Intel SGX). One or more software programs or libraries can be executed by the service TEE 306a. For the purposes of the present specification, the service TEE 306a refers to the secure environment (the TEE) as well as software executing inside the TEE that performs the described operations. In some embodiments, the service TEE 306a executes smart contract operations specified by encrypted client requests and outputs encrypted results associated with the smart contract operations. This functionality is described in greater detail with respect to FIG. 4 below.

**[049]** Node 304a also includes a key management TEE (KM TEE) 308a. In some embodiments, the KM TEE 308a is a secure application environment implemented using a TEE technology (e.g., Intel SGX). One or more software programs or libraries can be executed by the KM TEE 308a. For the purposes of the present specification, the KM TEE 308a refers to the secure environment (the TEE) as well as software executing inside the TEE that performs the described operations. In some embodiments, the KM TEE 308a obtains encryption keys from the key management center 310 as described in greater detail with respect to FIG. 4 below.

**[050]** The key management center 310 can generate, store, and maintain encryption keys. The key management center 310 can also authenticate identities of KM TEEs 308a-d and provide the encryption keys to the nodes 304a-d through a remote attestation and key

deployment process 320. In some embodiments, the key management can further provide encryption keys to clients for interacting with the nodes 304a-d. This functionality is described in greater detail with respect to FIG. 4 below. In some embodiments, the key management center 310 can be one or more servers or other computing devices in communication with one or more nodes of the blockchain network 302 over a communications network (not shown). The key management center 310 can also include one or more storage devices coupled to the key management center 310 or accessible over the communications network for storing the encryption keys and other data.

**[051]** In some cases, the key management center 310 operates to authenticate the identities of the KM TEEs 308a-d before performing encryption key deployments. For example, prior to providing the one or more encryption keys (described below) to the KM TEE 308a, the key management center 310 can verify the authenticity of the KM TEE 308a. This verification ensures that the software executed by the KM TEE 308a has not been tampered with after being provisioned. In some embodiments, the verification can include a remote attestation process 320, such as those described above.

**[052]** After the KM TEEs 308a-d obtain the one or more encryption keys from the key management center 310, the keys can be forwarded to service TEEs 306a-d to perform cryptographic operations. In some cases, although a KM TEE and service TEE pair (e.g., KM TEE 308a and service TEE 306a) operate on a single node (e.g., node 304a), they each have their own independent TEEs. As a result, information communicated between the KM TEEs 308a-d and service TEEs 306a-d is transmitted through untrusted area. In such cases, the KM TEEs 308a-d can authenticate the identities of the service TEEs 306a-d, for example by performing a local attestation process.

**[053]** Local attestation can allow an enclave to prove its identity or authenticity to another enclave within the same local platform. For example, a KM TEE 308a can send a challenge to verify the authenticity of the service TEE 306a. Upon receiving the challenge, the service TEE 306a can request a hardware (e.g., CPU) of node 304a to generate a report, which includes cryptographic proof that the service TEE 306a exists on the node 304a. The report can be provided to the KM TEE 308a to verify that the enclave report was generated on the same platform by node 304a. In some cases, the local attestation can be performed

based on a symmetric key system where only the KM TEE 308a verifying the report and the enclave hardware generating the report know the symmetric key, which is embedded in the hardware platform of node 304a.

**[054]** After the service TEE 306a-d are authenticated through local attestations, the KM TEE 308a-d can provide the one or more encryption keys to the service TEEs 306a-d. In some cases, the KM TEE 308a-d can provide the encryption keys in response to the authentication of the service TEE 306a-d, or can provide the keys in response to one or more requests by the service TEE 306a-d.

**[055]** The smart contract service logic 330 includes one or more smart contract definitions. Nodes 304a-304d execute particular operations from the smart contract service logic 330 (e.g., upon request of a client, as shown in FIG. 4). In some embodiments, the smart contract definitions in the smart contract service logic 330 include instructions to be executed by nodes of the blockchain network 302. The smart contract service logic 330 can include the smart contract definitions stored in one or more blockchains maintained by the blockchain network 302 (not shown).

**[056]** FIG. 4 is a diagram illustrating an example of a system 400 in accordance with implementations of this specification. As shown, the system 400 includes the node 304a (including the service TEE 306a and the KM TEE 308a), and the key management center 310 described with respect to FIG. 3. The system 400 also includes a client 480 communicatively coupled to the key management center 310.

**[057]** In operation, the system 400 can securely execute smart contract instructions and produce encrypted results of the operation (e.g., for inclusion in a blockchain). As discussed above, the key management center 310 can perform remote attestation to authenticate identity of the KM TEE 308a before trusting it with the encryption keys. After the KM TEE 308 is authenticated, the key management center 310 can provide an unseal private key 402, a root key 404 and a sign private key 406 to the KM TEE 308a of the Node 304a. The key management center 310 also hosts a seal public key 414 and a verification public key 416. The key management center 310 provides these keys to authorized clients to perform encryption and decryption of various data associated with the service TEE 306a, as described below.

**[058]** As shown, the key management center 310 provides the seal public key 414 to the client 480. In some cases, the key management center 310 authenticates the client 480 and only provides the seal public key 414 if the client 480 is authorized to access it. The key management center 310 can consult an internal or external permissions resource to make this determination. The seal public key 414 is associated with an unseal private key 402 provided to the KM TEE 308a. The seal public key 414 and the unseal private key 402 form a key pair, meaning that data encrypted with the seal public key 414 can be decrypted using the unseal private key 402.

**[059]** The client 480 identifies a requested contract operation 450, which is a smart contract operation to be executed by an Ethereum virtual machine (VM) 460 deployed in the service TEE 306a. In some cases, the smart contract operations 450 include one or more instructions encoded in a smart contract programming language for execution by a VM operable to execute instruction in that language. The smart contract operations 450 can include an execution state for the smart contract associated with the request contract operation 450. During execution of a smart contract, multiple nodes of a blockchain network execute each instruction of the smart contract individually, and produce a result indicating an execution state of the smart contract after the completion of the instruction. The execution state can include data associated with the smart contract. Each executed instruction of the contract can change the contents of the data (e.g., to store a value to be used by a later instruction in the smart contract). After execution of an instruction of the smart contract, the nodes of the blockchain network reach a consensus on the new execution state after execution of the instruction. This consensus process is performed for each instruction executed in a smart contract, leading to a consensus as to the execution path of the smart contract and, ultimately, as to the final result of the execution.

**[060]** At 452, the client 480 encodes (or seals) the requested contract operation 450 in a digital envelope 454 for transmission to the service TEE 306a executed by the Node 304a. For example, the client 480 generates a temporary symmetric key 408 and encrypts the requested contract operation 450 using the key 408. The client 480 then encrypts the temporary symmetric key 408 using the seal public key 414, and concatenates the encrypted contract operation 450 and the encrypted key 408 to produce the digital envelope 454.

[061] The client 480 transmits the digital envelope 454 to the node 304a, where it is provided to the service TEE 306a. In some cases, the client 480 can send the digital envelope 454 to multiple nodes 304a-d to request processing of the requested contract operation 450. In some cases, the client 480 can send digital envelopes created using seal public keys specific to the particular nodes. The client 480 can also broadcast the digital envelope 454 to the nodes 304a-d in cases where the same seal public key 414 and unseal private key 402 are associated with all the nodes 304a-d.

[062] The service TEE 306a receives the digital envelope 454 from the client 480 and recovers the requested contract operation 450 from the digital envelope 454. As shown, the service TEE 306a decodes the digital envelope 454 using the unseal private key 402 obtained from the KM TEE 308a. In some cases, the service TEE 306a decrypts (unseals) the temporary symmetric key 408 using the unseal private key 402 (at 456), and then decrypts the requested contract operation 450 using the temporary symmetric key 408 (at 458).

[063] The service TEE 306a then executes the requested contract operation 450 using a VM 460 deployed in the service TEE 306a. In some embodiments, the VM 460 can be a VM configured to execute instructions of a smart contract programming language, such as an Ethereum VM, or other type of VM. In some cases, the VM 460 can access resources external the service TEE 306a during execution of the operation 450, such as, for example, external servers, a blockchain, a database, or other resources indicated by the operation 450. In some embodiments, accessing external resources can be restricted or denied, such that the entirety of the execution of the operation depends only on data stored in the service TEE 306a (such as the smart contract state). This type of restriction can further reduce the possibility of tampering with the execution of the operation 450.

[064] The execution of the operation 450 by the VM 460 can produce one or more results. In some cases, the results can include an execution state of the smart contract after executing the operation 450, as described above. At 462, the result of the smart contract operation 450 is encrypted by the service TEE 306a using a contract key 412. The contract key 412 is derived (at 410) from a root key 404 based on a key derivation function (KDF). In some examples, the KDF can be performed based on iterative hash algorithms, such as HMAC-based extract-and-expand key derivation function (HKDF) or pseudo-random

function (PRF). The contract key can be provided by the KM TEE 308a to the service TEE306a. In some embodiments, the root key 404 can be a symmetric encryption key associated with the node 304a. The root key 404 can also include one or more subkeys that can be derived from the root key 404. The contract key 412 can be one of these subkeys. In some cases, the root key 404 itself can be used to encrypt the result at 462.

[065] After encrypting the result, the service TEE 308a, at 464, signs the encrypted result using a sign private key 406 provided by the KM TEE 308a to the service TEE 306a, so as to produce a signed result 466. This can allow later verification of the signed result by a third party (for example, a client), using the verification public key 416 (correspondingly paired to the sign private key 406) maintained by the key management center 310. In some cases, signing the encrypted result by the sign private key 406 can include encrypting the encrypted result together with the contract key 412 used to encrypt the result. In such a case, a third party holding the verification public key 416 can decrypt the the contract key 412 first, and further use the contract key 412 to decrypt the result.

[066] In some cases, the service TEE 306a can store the signed result 466 in a blockchain. As described above, a third party holding the verification public key 416 can use the key to decrypt the result 466 in order to inspect. For example, the client 480 can retrieve the verification public key 416 from the key management center 310 (e.g., subject to authentication as previously described), and can access and decrypt the signed result 466 using the verification public key 416. The client 480 can then request that the next operation in the smart contract be executed by the service TEE 306a, an can include the requested next operation and the execution state of the smart contract (from the decrypted signed result 466) in the digital envelope sent to the service TEE 306a.

[067] FIG. 5 depicts an example of a process that can be executed in accordance with embodiments of this specification. At 502, a blockchain node (e.g., 304a) participating in a blockchain network (e.g., 302) receives a request to execute one or more software instructions in a service TEE hosted by the blockchain node, wherein the request is encrypted by a public key associated with the service TEE.

[068] At 504, the blockchain node decrypts the request with a first private key associated with the service TEE, wherein the first private key is paired with the public key.

[069] At 506, the blockchain node executes the one or more software instructions to produce an execution result.

[070] At 508, the blockchain node encrypts the execution result with a client encryption key associated with the service TEE to produce an encrypted result.

[071] At 510, the blockchain node signs the encrypted result using a second private key associated with the TEE to produce a signed encrypted result.

[072] In some cases, the public key is a first public key, and the client encryption key is one of a second public key or a symmetric key derived from a root key based on a key derivation function.

[073] In some cases, the blockchain node further hosts a key management TEE that stores one or more of the first private key, the second private key, and the key management TEE provides the first private key, the second private key, and the root key to the service TEE after an identity of the service TEE is authenticated based on performing a local attestation initiated by the key management TEE.

[074] In some cases, the first private key, the second private key, and the root key are generated by a key management center and are provided to the key management TEE after an identity of the key management TEE is authenticated based on performing a remote attestation initiated by the key management center.

[075] In some cases, the first private key and the root key are provided by the key management TEE to the service TEE in response to a rebooting operation of the service TEE.

[076] In some cases, the one or more software instructions are associated with a smart contract, and the root key is selected from a plurality of the root keys stored in the key management TEE based on a state of the smart contract.

[077] In some cases, the first public key is generated by the key management center and provided to a client for encrypting the request.

[078] In some cases, the request received by the blockchain node further includes using the client encryption key to encrypts the one or more software instructions.

[079] In some cases, decrypting the request with the first private key further comprises: decrypting the client encryption key with the first private key; and decrypting the one or more software instructions with the client encryption key.

[080] FIG. 6 depicts examples of modules of an apparatus 600 in accordance with embodiments of this specification. The apparatus 600 can be an example embodiment of a blockchain node executing within a blockchain network. The apparatus 600 can correspond to the embodiments described above, and the apparatus 600 includes the following: a receiving module 602 that receives a request to execute one or more software instructions in a service TEE hosted by the blockchain node, wherein the request is encrypted by a public key associated with the service TEE; a decrypting module 604 that decrypts the request with a first private key associated with the service TEE, wherein the first private key is paired with the public key; an executing module 606 that executes the one or more software instructions to produce an execution result; an encrypting module 608 that encrypts the execution result with a client encryption key associated with the service TEE to produce an encrypted result; a signing module 610 that signs the encrypted result using a second private key associated with the TEE to produce a signed encrypted result.

[081] The system, apparatus, module, or unit illustrated in the previous 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 embodiment 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.

[082] For an embodiment process of functions and roles of each module in the apparatus, references can be made to an embodiment process of corresponding steps in the previous method. Details are omitted here for simplicity.

[083] Because an apparatus embodiment basically corresponds to a method embodiment, for related parts, references can be made to related descriptions in the method embodiment. The previously described apparatus embodiment is merely an example. The modules described as separate parts may or may not be physically separate, and parts displayed as

modules may or may not be physical modules, may be located in one position, or may be distributed on a number of network modules. Some or all of the modules can be selected based on actual demands to achieve the objectives of the solutions of the specification. A person of ordinary skill in the art can understand and implement the embodiments of the present application without creative efforts.

**[084]** Referring again to FIG. 6, it can be interpreted as illustrating an internal functional module and a structure of a blockchain node executing within a blockchain network and functioning as an execution body. An execution body in essence can be an electronic device, and the electronic device includes the following: one or more processors; and a memory configured to store an executable instruction of the one or more processors.

**[085]** The techniques described in this specification produce one or more technical effects. For example, the described techniques enables parties to a consortium blockchain network to verify that blockchain nodes in the network that are responsible for executing smart contract operations and arriving at a consensus as to the results of those operations have not been compromised by an attacker. This verification has the effect of reducing the preventing or reducing the likelihood of an attacker taking control of one or more of the blockchain nodes and successfully tampering with the execution of the smart contract operations or the consensus process, leading a more secure embodiment of a consortium blockchain network that is more resistant to attacks.

**[086]** Described embodiments of the subject matter can include one or more features, alone or in combination. One embodiment includes a computer-implemented method comprising the actions of receiving, by a blockchain node participating in a blockchain network, a request to execute one or more software instructions in a service TEE hosted by the blockchain node, wherein the request is encrypted by a public key associated with the service TEE; decrypting, by the blockchain node in the service TEE, the request with a first private key associated with the service TEE, wherein the first private key is paired with the public key; in response to decrypting the request, executing, by the blockchain node in the service TEE, the one or more software instructions to produce an execution result; encrypting, by the blockchain node in the service TEE, the execution result with a client encryption key associated with the service TEE to produce an encrypted result; and signing,

by the blockchain node in the TEE, the encrypted result using a second private key associated with the TEE to produce a signed encrypted result.

[087] The foregoing and other described embodiments can each, optionally, include one or more of the following features:

[088] A first feature, combinable with any of the following features, specifies that the public key is a first public key, and the client encryption key is one of a second public key or a symmetric key derived from a root key based on a key derivation function.

[089] A second feature, combinable with any of the previous or following features, specifies that the blockchain node further hosts a key management TEE that stores one or more of the first private key, the second private key, and wherein the key management TEE provides the first private key, the second private key, and the root key to the service TEE after an identity of the service TEE is authenticated based on performing a local attestation initiated by the key management TEE.

[090] A third feature, combinable with any of the previous or following features, specifies that the first private key, the second private key, and the root key are generated by a key management center and are provided to the key management TEE after an identity of the key management TEE is authenticated based on performing a remote attestation initiated by the key management center.

[091] A fourth feature, combinable with any of the previous or following features, specifies that the first private key and the root key are provided by the key management TEE to the service TEE in response to a rebooting operation of the service TEE.

[092] A fifth feature, combinable with any of the previous or following features, specifies that the one or more software instructions are associated with a smart contract, and wherein the root key is selected from a plurality of the root keys stored in the key management TEE based on a state of the smart contract.

[093] A sixth feature, combinable with any of the previous or following features, specifies that the first public key is generated by the key management center and provided to a client for encrypting the request.

[094] A seventh feature, combinable with any of the previous or following features, specifies that the request received by the blockchain node further includes using the client encryption key to encrypts the one or more software instructions.

[095] An eighth feature, combinable with any of the previous or following features, specifies that decrypting the request with the first private key further comprises: decrypting the client encryption key with the first private key; and decrypting the one or more software instructions with the client encryption key.

[096] A ninth feature, combinable with any of the previous or following features, specifies that key management center stores a verification public key that corresponds to the second private key and provides the verification public key to the client for verifying the signed encrypted result.

[097] Embodiments of the subject matter and the actions and operations described in this specification can be implemented in digital electronic circuitry, in tangibly-embodied computer software or firmware, in computer hardware, including the structures disclosed in this specification and their structural equivalents, or in combinations of one or more of them. Embodiments of the subject matter described in this specification can be implemented as one or more computer programs, e.g., one or more modules of computer program instructions, encoded on a computer program carrier, for execution by, or to control the operation of, data processing apparatus. For example, a computer program carrier can include one or more computer-readable storage media that have instructions encoded or stored thereon. The carrier may be a tangible non-transitory computer-readable medium, such as a magnetic, magneto optical, or optical disk, a solid state drive, a random access memory (RAM), a read-only memory (ROM), or other types of media. Alternatively, or in addition, the carrier may be an artificially generated propagated signal, e.g., a machine-generated electrical, optical, or electromagnetic signal that is generated to encode information for transmission to suitable receiver apparatus for execution by a data processing apparatus. The computer storage medium can be or be part of a machine-readable storage device, a machine-readable storage substrate, a random or serial access memory device, or a combination of one or more of them. A computer storage medium is not a propagated signal.

**[0098]** A computer program, which may also be referred to or described as a program, software, a software application, an app, a module, a software module, an engine, a script, or code, can be written in any form of programming language, including compiled or interpreted languages, or declarative or procedural languages; and it can be deployed in any form, including as a stand-alone program or as a module, component, engine, subroutine, or other unit suitable for executing in a computing environment, which environment may include one or more computers interconnected by a data communication network in one or more locations.

**[0099]** A computer program may, but need not, correspond to a file in a file system. A computer program can be stored in a portion of a file that holds other programs or data, e.g., one or more scripts stored in a markup language document, in a single file dedicated to the program in question, or in multiple coordinated files, e.g., files that store one or more modules, sub programs, or portions of code.

**[0100]** Processors for execution of a computer program include, by way of example, both general- and special-purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive the instructions of the computer program for execution as well as data from a non-transitory computer-readable medium coupled to the processor.

**[0101]** The term “data processing apparatus” encompasses all kinds of apparatuses, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. Data processing apparatus can include special-purpose logic circuitry, e.g., an FPGA (field programmable gate array), an ASIC (application specific integrated circuit), or a GPU (graphics processing unit). The apparatus can also include, in addition to hardware, code that creates an execution environment for computer programs, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them.

**[0102]** The processes and logic flows described in this specification can be performed by one or more computers or processors executing one or more computer programs to perform operations by operating on input data and generating output. The processes and logic flows

can also be performed by special-purpose logic circuitry, e.g., an FPGA, an ASIC, or a GPU, or by a combination of special-purpose logic circuitry and one or more programmed computers.

**[0103]** Computers suitable for the execution of a computer program can be based on general or special-purpose microprocessors or both, or any other kind of central processing unit. Generally, a central processing unit will receive instructions and data from a read only memory or a random access memory or both. Elements of a computer can include a central processing unit for executing instructions and one or more memory devices for storing instructions and data. The central processing unit and the memory can be supplemented by, or incorporated in, special-purpose logic circuitry.

**[0104]** Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to one or more storage devices. The storage devices can be, for example, magnetic, magneto optical, or optical disks, solid state drives, or any other type of non-transitory, computer-readable media. However, a computer need not have such devices. Thus, a computer may be coupled to one or more storage devices, such as, one or more memories, that are local and/or remote. For example, a computer can include one or more local memories that are integral components of the computer, or the computer can be coupled to one or more remote memories that are in a cloud network. Moreover, a computer can be embedded in another device, e.g., a mobile telephone, a personal digital assistant (PDA), a mobile audio or video player, a game console, a Global Positioning System (GPS) receiver, or a portable storage device, e.g., a universal serial bus (USB) flash drive, to name just a few.

**[0105]** Components can be “coupled to” each other by being commutatively such as electrically or optically connected to one another, either directly or via one or more intermediate components. Components can also be “coupled to” each other if one of the components is integrated into the other. For example, a storage component that is integrated into a processor (e.g., an L2 cache component) is “coupled to” the processor.

**[0106]** To provide for interaction with a user, embodiments of the subject matter described in this specification can be implemented on, or configured to communicate with, a computer having a display device, e.g., a LCD (liquid crystal display) monitor, for displaying information to the user, and an input device by which the user can provide input to the

computer, e.g., a keyboard and a pointing device, e.g., a mouse, a trackball or touchpad. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback, e.g., visual feedback, auditory feedback, or tactile feedback; and input from the user can be received in any form, including acoustic, speech, or tactile input. In addition, a computer can interact with a user by sending documents to and receiving documents from a device that is used by the user; for example, by sending web pages to a web browser on a user's device in response to requests received from the web browser, or by interacting with an app running on a user device, e.g., a smartphone or electronic tablet. Also, a computer can interact with a user by sending text messages or other forms of message to a personal device, e.g., a smartphone that is running a messaging application, and receiving responsive messages from the user in return.

**[0107]** This specification uses the term “configured to” in connection with systems, apparatus, and computer program components. For a system of one or more computers to be configured to perform particular operations or actions means that the system has installed on it software, firmware, hardware, or a combination of them that in operation cause the system to perform the operations or actions. For one or more computer programs to be configured to perform particular operations or actions means that the one or more programs include instructions that, when executed by data processing apparatus, cause the apparatus to perform the operations or actions. For special-purpose logic circuitry to be configured to perform particular operations or actions means that the circuitry has electronic logic that performs the operations or actions.

**[0108]** While this specification contains many specific embodiment details, these should not be construed as limitations on the scope of what is being claimed, which is defined by the claims themselves, but rather as descriptions of features that may be specific to particular embodiments. Certain features that are described in this specification in the context of separate embodiments can also be realized in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiments can also be realized in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially be claimed as such, one or more features from a claimed combination can in

some cases be excised from the combination, and the claim may be directed to a subcombination or variation of a subcombination.

**[0109]** Similarly, while operations are depicted in the drawings and recited in the claims in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. In certain circumstances, multitasking and parallel processing may be advantageous. Moreover, the separation of various system modules and components in the embodiments described above should not be understood as requiring such separation in all embodiments, and it should be understood that the described program components and systems can generally be integrated together in a single software product or packaged into multiple software products.

**[0110]** Particular embodiments of the subject matter have been described. Other embodiments are within the scope of the following claims. For example, the actions recited in the claims can be performed in a different order and still achieve desirable results. As one example, the processes depicted in the accompanying figures do not necessarily require the particular order shown, or sequential order, to achieve desirable results. In some cases, multitasking and parallel processing may be advantageous.

## CLAIMS

What is claimed is:

1. A computer-implemented method for securely executing smart contract operations in a trusted execution environment (TEE), the method comprising:
  - receiving, by a blockchain node participating in a blockchain network, a request to execute one or more software instructions in a service TEE hosted by the blockchain node, wherein the request is encrypted by a public key associated with the service TEE;
  - decrypting, by the blockchain node in the service TEE, the request with a first private key associated with the service TEE, wherein the first private key is paired with the public key;
  - in response to decrypting the request, executing, by the blockchain node in the service TEE, the one or more software instructions to produce an execution result;
  - encrypting, by the blockchain node in the service TEE, the execution result with a client encryption key associated with the service TEE to produce an encrypted result; and
  - signing, by the blockchain node in the TEE, the encrypted result using a second private key associated with the TEE to produce a signed encrypted result.
2. The computer-implemented method of claim 1, wherein the public key is a first public key, and the client encryption key is one of a second public key or a symmetric key derived from a root key based on a key derivation function.
3. The computer-implemented method of claim 2, wherein the blockchain node further hosts a key management TEE that stores one or more of the first private key, the second private key, and wherein the key management TEE provides the first private key, the second private key, and the root key to the service TEE after an identity of the service TEE is authenticated based on performing a local attestation initiated by the key management TEE.
4. The computer-implemented method of any one of the preceding claims, wherein the first private key, the second private key, and the root key are generated by a key management center and are provided to the key management TEE after an identity of the key management

TEE is authenticated based on performing a remote attestation initiated by the key management center.

5. The computer-implemented method of any one of the preceding claims, wherein the first private key and the root key are provided by the key management TEE to the service TEE in response to a rebooting operation of the service TEE.

6. The computer-implemented method of any one of the preceding claims, wherein the one or more software instructions are associated with a smart contract, and wherein the root key is selected from a plurality of the root keys stored in the key management TEE based on a state of the smart contract.

7. The computer-implemented method of any one of the preceding claims, wherein the first public key is generated by the key management center and provided to a client for encrypting the request.

8. The computer-implemented method of any one of the preceding claims, wherein the request received by the blockchain node further includes using the client encryption key to encrypts the one or more software instructions.

9. The computer-implemented method of claim 8, wherein decrypting the request with the first private key further comprises:

decrypting the client encryption key with the first private key; and  
decrypting the one or more software instructions with the client encryption key.

10. The computer-implemented method of any one of the preceding claims, wherein the key management center stores a verification public key that corresponds to the second private key and provides the verification public key to the client for verifying the signed encrypted result.

11. A system for securely executing smart contract operations in a trusted execution environment (TEE), comprising:  
one or more processors; and  
one or more computer-readable memories coupled to the one or more processors and having instructions stored thereon which are executable by the one or more processors to perform the method of any of claims 1-10.
12. An apparatus for securely executing smart contract operations in a trusted execution environment (TEE), the apparatus comprising a plurality of modules for performing the method of any one of claims 1-10.