Methods, Apparatuses and Systems for Obtaining Data Authorization

This application claims priority to the Chinese patent application titled "Method, Apparatus and System for Obtaining Data Authorization" filed with the China National Intellectual Property Administration on December 23, 2021 with application number 202111585885.2, the entire contents of which are incorporated herein by reference.

Technical Field

[01] One or more example embodiments of the present disclosure relate to the field of security technologies, and in particular, to methods, apparatuses and systems for obtaining data authorization.

Background Art

[02] Confidential computing refers to a data provider uploading data to a central server which performs computations on the data in encrypted form without requiring participation from the data provider. Confidential computing can be applied in many scenarios. For example, a data provider can provide data in encrypted form to a trading center. The trading center can compute valuable results in encrypted form and sell the results to data requesters. The data provider and trading center can then share the revenue generated.

[03] In current confidential computing solutions, data providers usually cannot control the computation logic used at the central server. However, data providers may only want data to be used for machine learning training and other logics that leak little information, rather than logics like quantiles and segmentation that leak more information. In addition, since the central server may intentionally use intentionally constructed and different computation logic to infer the original data from the computation results, there is a need for a solution that can constrain the computation logic used at the trusted computing center.

Summary of the Invention

[04] One or more example embodiments of the present disclosure describe methods, apparatuses and systems for obtaining data authorization that can constrain the computation logic used at a trusted computing center, thus ensuring the security of data providers' data.

[05] In a first aspect, a method for obtaining data authorization is provided, comprising:

[06] Any first computing node among the plurality of trusted computing nodes receiving an authentication request from the first provider;

[07] Returning authentication information to the first provider, wherein the authentication information comprises at least a first code hash of a first computation logic running on the first computing node;

[08] Receiving a channel establishment request sent by the first provider after determining that the first computing node has passed trusted authentication and the first code hash has passed correctness verification;

[09] Establishing a first trusted channel with the first provider based on the channel establishment request;

[10] Obtaining, via the first trusted channel, a target encryption key from the respective encryption keys corresponding to the plurality of encrypted shards, thereby obtaining authorization to compute the target encrypted shard corresponding to the target encryption key based on the first computation logic.

[11] In a second aspect, a method for obtaining data authorization is provided, comprising:

[12] Respective proxy nodes obtaining respective encryption keys corresponding to the plurality of encrypted shards from the first provider;

[13] Any first proxy node among the respective proxy nodes sending an authentication request to a corresponding first computing node and receiving authentication information returned by the first computing node, wherein the authentication information comprises at least a first code hash of a first computation logic running on the first computing node;

[14] The first proxy node performing trusted authentication on the first computing node based on the authentication information, and judging whether the first code hash has been verifiably stored;

[15] In response to the trusted authentication passing and the first code hash having been verifiably stored, the first proxy node establishing a first trusted channel with the first computing node;

[16] The first proxy node sending a target encryption key received by it to the first computing node via the first trusted channel;

[17] The first computing node obtaining authorization to compute the target encrypted shard corresponding to the target encryption key based on the first computation logic.

[18] In a third aspect, a method for obtaining data authorization is provided, comprising:

[19] Obtaining a target encryption key from the respective encryption keys corresponding to the plurality of encrypted shards from the first provider;

[20] Sending an authentication request to a corresponding first computing node;

[21] Receiving authentication information returned by the first computing node, wherein the authentication information comprises at least a first code hash of a first computation logic running on the first computing node;

[22] Performing, based on the authentication information, trusted authentication on the first computing node and judging whether the first code hash has been verifiably stored;

[23] In response to the trusted authentication passing and the first code hash having been verifiably stored, establishing a first trusted channel with the first computing node;

[24] Sending, via the first trusted channel, the target encryption key to the first computing node; thereby the first computing node obtaining authorization to compute the target encrypted shard corresponding to the target encryption key based on the first computation logic.

[25] In a fourth aspect, an apparatus for obtaining data authorization is provided, comprising:

[26] A receiving unit for receiving an authentication request from the first provider;

[27] A sending unit for returning authentication information to the first provider, wherein the authentication information comprises at least a first code hash of a first computation logic running on the first computing node;

[28] wherein the receiving unit is further configured to receive a channel establishment request sent by the first provider after determining that the first computing node has passed trusted authentication and the first code hash has passed correctness verification;

[29] An establishing unit for establishing, based on the channel establishment request, a first trusted channel with the first provider;

[30] wherein the receiving unit is further configured to obtain, via the first trusted channel, a target encryption key from the respective encryption keys corresponding to the plurality of encrypted shards, thereby obtaining authorization to compute the target encrypted shard corresponding to the target encryption key based on the first computation logic.

[31] In a fifth aspect, a system for obtaining data authorization is provided, comprising:

[32] Respective proxy nodes for obtaining respective encryption keys corresponding to the plurality of encrypted shards from the first provider;

[33] Any first proxy node among respective proxy nodes for sending an authentication request to a corresponding first computing node and receiving authentication information returned by the first computing node, wherein the authentication information comprises at least a first code hash of a first computation logic running on the first computing node;

[34] wherein the first proxy node is further configured to perform, based on the authentication information, trusted authentication on the first computing node and judge whether the first code hash has been verifiably stored;

[35] wherein the first proxy node further is configured to, in response to the trusted authentication passing and the first code hash having been verifiably stored, establish a first trusted channel with the first computing node;

[36] wherein the first proxy node is further configured to send a target encryption key received by it to the first computing node via the first trusted channel;

[37] The first computing node for obtaining authorization to compute the target encrypted shard corresponding to the target encryption key based on the first computation logic.

[38] In a sixth aspect, an apparatus for obtaining data authorization is provided, comprising:

[39] An obtaining unit, configured to obtain a target encryption key from the respective encryption keys corresponding to the plurality of encrypted shards from the first provider;

[40] A sending unit, configured to send an authentication request to a corresponding first computing node;

[41] A receiving unit, configured to receive authentication information returned by the first computing node, wherein the authentication information comprises at least a first code hash of a first computation logic running on the first computing node;

[42] A determination unit, configured to perform trusted authentication on the first computing node based on the authentication information, and determine whether the first code hash is verifiably stored;

[43] An establishment unit, configured to establish a first trusted channel with the first

computing node when the trusted authentication passes and the first code hash is verifiably stored;

[44] The sending unit is further configured to send the target encryption key to the first computing node through the first trusted channel; thereby the first computing node obtains authorization to compute the target encrypted shard corresponding to the target encryption key based on the first computation logic.

[45] In a seventh aspect, a computer storage medium is provided, storing a computer program which, when executed by a computer, causes the computer to perform the methods according to the first, second or third aspect.

[46] In an eighth aspect, a computing device is provided, comprising a memory and a processor, wherein the memory stores executable code which, when executed by the processor, implements the methods according to the first, second or third aspect.

[47] The method, apparatus and system for obtaining data authorization disclosed in one or more example embodiments of the present disclosure propose two ways to constrain the computation logic used at a trusted computing center. In one approach, the data provider reviews the computation logic in advance and stores the code hash of the reviewed computation logic. Afterwards, the data provider can obtain the code hash of the computation logic running on the trusted computing node, and provide the encryption key of the data to the trusted computing node to achieve data authorization if the obtained code hash is consistent with the stored code hash. In the other approach, the trusted computing node verifiably stores the code hash of the computation logic running on it in advance at a verifiable storage center. Afterwards, the proxy node of the trusted computing node can obtain the code hash of the computation logic running on the trusted computing node, and provide the encryption key of the data obtained in advance from the data provider to achieve data authorization if the obtained code hash is consistent with the verifiably stored code hash. In this approach, the data provider can obtain the code hash from the verifiable storage center and review the corresponding computation logic for legitimacy based on the code hash.

Brief Description of Drawings

[48] In order to more clearly illustrate the technical solutions in the embodiments of the present disclosure, the drawings to be used in the description of the embodiments are briefly introduced below. Apparently, the drawings in the following descriptions are only some embodiments of the present disclosure. For those of ordinary skill in the art, other drawings can be obtained without creative efforts based on these drawings.

[49] Figure 1 shows a schematic diagram of TECC according to an embodiment;

[50] Figure 2 shows a schematic diagram of an application scenario disclosed in an

embodiment of the present disclosure;

[51] Figure 3 shows a schematic diagram of another application scenario disclosed in an embodiment of the present disclosure;

[52] Figure 4 shows an interaction diagram of a method for uploading encrypted data according to an embodiment;

[53] Figure 5 shows an interaction diagram of a method for obtaining data authorization according to an embodiment;

[54] Figure 6 shows an interaction diagram of a method for obtaining data authorization according to another embodiment;

[55] Figure 7 shows a schematic diagram of an apparatus for obtaining data authorization according to an embodiment;

[56] Figure 8 shows a schematic diagram of a system for obtaining data authorization according to an embodiment;

[57] Figure 9 shows a schematic diagram of an apparatus for obtaining data authorization according to another embodiment.

Detailed Description

[58] The solutions provided in the present disclosure are described below with reference to the accompanying drawings.

[59] The solutions proposed in the embodiments of the present disclosure propose a scheme to constrain the computation logic used by the center when computing data providers' data using Trusted Cryptographic Computing (TECC).

[60] TECC is a secure and efficient confidential computing method that can compute a common result for multiple data providers without leaking any party's data. Trusted confidential computing combines both system security and cryptography technologies to better balance security and performance than using only one technology, ultimately achieving the ideal state of "sufficient security and very fast performance".

[61] Cryptography can be used to construct mathematically provably secure schemes, but such schemes tend to have poor computational performance. Due to inherent properties of mathematics, it is difficult to improve to an ideal state that meets the requirements of most scenarios. Therefore, we seek a "sufficient but not strictly mathematically provable" solution. Trusted security technologies represented by TEE (Trusted Execution Environment) have significantly better security than common

operating systems because they rely on less code for security. This is a viable approach. However, practice has proven that TEEs can still be compromised. TECC combines both approaches, so that when the TEE is compromised, security is still guaranteed, meeting the requirements of the vast majority of scenarios. On the other hand, TECC also inherits the high performance characteristic of trusted technologies, and with reasonable resource expansion, can achieve performance approximating plaintext computation.

[62] Figure 1 shows a schematic diagram of TECC according to an embodiment. In Figure 1, the data provider U randomly splits the data into multiple data shards locally: U1, U2 and U3. 2) The data provider U establishes secure channels with multiple TEEs and sends each data shard to a separate TEE. For example, U1, U2 and U3 are sent to TEEX, TEEY and TEEZ respectively. Other data providers can also send their respective data shards to separate TEEs. 3) After acquiring all data shards from data providers, each TEE executes MPC (Secure Multi-Party Computation) protocols based on secret sharing to complete the computation.

[63] It should be noted that in TECC, the data provider can use TEE technology to ensure their data only exist within the TEE, and the TEE's host, owner, etc. cannot access the plaintext data (if the TEE is not compromised). On the other hand, each TEE only ever touches a data shard, so even if an attacker compromises a TEE and steals or modifies long-term, they cannot obtain valid information. In practice, this level of defense is almost unbreakable. TECC's high performance mainly stems from very little public network interaction, generally only once. With data pre-uploaded, public network interaction may not even be needed. Another reason is that TECC supports high concurrency, allowing it to achieve near-plaintext performance with reasonable resource expansion.

[64] Figure 2 shows a schematic diagram of an application scenario disclosed in an embodiment of the present disclosure. In Figure 2, the data provider U can upload encrypted data to the trusted computing center, and the trusted computing center can compute the encrypted data based on a computation logic. Here, the data provider U can be implemented as any device, platform, server or cluster with computing and processing capabilities. The trusted computing center can include a data storage center and three trusted computing nodes, denoted as: TEEX, TEEY and TEEZ. In addition, the trusted computing center can also include a code storage center and a management center. The code storage center is used to store the computation logic operated on at each trusted computing node. The management center is used to manage each trusted computing node.

[65] Specifically, the data provider U can split the corresponding private data into three private shards denoted as: U1, U2 and U3, and generate three random encryption keys corresponding to the three private shards: K1, K2 and K3. Afterwards, the data provider U can use the three encryption keys to encrypt the three private shards and obtain three

encrypted shards denoted as: ENC(K1, U1), ENC(K2, U2), ENC(K3, U3), and upload the three encrypted shards to the data storage center of the trusted computing center respectively.

[66] After uploading each encrypted shard, the data provider U can review any computation logic, save the code hash of the reviewed computation logic, and obtain the code hashes of the computation logics running on TEEX, TEEY and TEEZ respectively. If the obtained code hashes are consistent with the saved code hashes, the data provider U provides the encryption keys K1, K2 and K3 to TEEX, TEEY and TEEZ respectively, so that TEEX, TEEY and TEEZ obtain the authorization to compute U1, U2 and U3 based on their respective running computation logics. That is, each TEE obtains the corresponding data authorization.

[67] In addition, TEEX, TEEY and TEEZ can also use their respective private keys to encrypt K1, K2 and K3 respectively, obtaining Seal(K1), Seal(K2) and Seal(K3), and store their respective encrypted encryption keys in the data storage center for subsequent joint computation.

[68] It should be noted that since the data provider U provides the encryption keys to each TEE only when the obtained code hashes are consistent with the saved code hashes, and the computation logic corresponding to the saved code hash has been reviewed by the data provider U, the computation logic used at the trusted computing center can be constrained or controlled.

[69] Likewise, other data providers can also upload their respective encrypted shards to the trusted computing center, and provide the corresponding encryption keys to each TEE if the obtained code hashes are consistent with the saved code hashes. This is not repeated here.

[70] In addition, it should be understood that Figure 2 is just an exemplary illustration. In actual applications, the number of TEEs can also be two or more, and accordingly, the data can be split into two private shards or more. Also, the number of encryption keys can be different from the number of TEEs, but equal to the number of TEE role categories, so that TEEs of the same role category share the same encryption key.

[71] Figure 3 shows a schematic diagram of another application scenario disclosed in an embodiment of the present disclosure. In Figure 3, the data provider U can upload encrypted data to the trusted computing center, and the trusted computing center can compute on the encrypted data based on a computation logic. Here, the data provider U can be implemented as any device, platform, server or cluster with computing and processing capabilities. The trusted computing center can include a data storage center and three trusted computing nodes and their respective proxy nodes. The three trusted computing nodes are denoted as TEEX, TEEY and TEEZ, and their corresponding proxy nodes are denoted as TEEA, TEEB and TEEC respectively. In addition, the

trusted computing center can also include a code storage center and a management center. The code storage center is used to store the computation logic operated on at each trusted computing node. The management center is used to manage each trusted computing node.

[72] Specifically, the data provider U can split the corresponding private data into three private shards denoted as U1, U2 and U3, and generate three random encryption keys corresponding to the three private shards: K1, K2 and K3. Afterwards, the data provider U can use the three encryption keys to encrypt the three private shards and obtain three encrypted shards denoted as ENC(K1, U1), ENC(K2, U2), ENC(K3, U3), and upload the three encrypted shards to the data storage center of the trusted computing center respectively.

[73] In addition, the data provider U can also send the encryption keys K1, K2 and K3 to TEEA, TEEB and TEEC respectively. TEEA, TEEB and TEEC can then use their respective private keys to encrypt K1, K2 and K3 respectively, obtaining Seal(K1), Seal(K2) and Seal(K3), and store their respective encrypted encryption keys in the data storage center for subsequent authorization of the corresponding trusted computing nodes. TEEX, TEEY and TEEZ can verifiably store the code hashes of the computation logics running on them at a verifiable storage center in advance.

[74] Afterwards, each proxy node can obtain from the corresponding trusted computing node the code hash of the running computation logic and its verifiably stored hash. If the running code hash matches the verifiably stored hash, the proxy node provides the encryption keys K1, K2 and K3 to TEEX, TEEY and TEEZ respectively, so that TEEX, TEEY and TEEZ obtain the authorization to compute U1, U2 and U3 based on their respective running computation logics. That is, each trusted computing node obtains the corresponding data authorization.

[75] The data provider U can obtain the verifiably stored code hashes of each trusted computing node from the verifiable storage center at any time, and then review the corresponding computation logic pulled from the code storage center based on the obtained code hashes. That is, the data provider U reviews the computation logic used at the trusted computing center afterwards, which also enables constraint or control over the computation logic used at the trusted computing center. Moreover, this post hoc review avoids frequent interaction between the data provider and the trusted computing center.

[76] Similarly, other data providers can also upload their respective encrypted shards to the trusted computing center, and provide the corresponding encryption keys to each trusted computing node if the obtained code hashes match the verifiably stored hashes. This is not repeated here.

[77] It should be understood that Figure 3 is just an exemplary illustration. In actual

applications, the number of trusted computing nodes can also be two or more. Accordingly, the data can also be split into two private shards or more. In addition, the number of proxy nodes does not have to equal the number of trusted computing nodes. For example, multiple trusted computing nodes can correspond to one proxy node. Finally, the trusted computing node and proxy node can be merged into one, so that the data provider U performs the verification of the verifiably stored code hash, etc. This is not limited herein.

[78] In summary, in the above two scenarios, the computation logic used at the trusted computing center can be constrained and controlled, thus ensuring the security of the data providers' data.

[79] The above two application scenarios show that the data provider authorizes the encrypted data uploaded in advance. Therefore, the method of the data provider uploading encrypted data is explained first below.

[80] Figure 4 shows an interaction diagram of a method for uploading encrypted data according to an embodiment. In Figure 4, the method can comprise at least the following steps:

[81] Step 402: The first provider splits the first private data to be stored into multiple private shards.

[82] In one example, the number of private shards can be determined based on the number of trusted computing nodes involved in secure multi-party computation. For example, in the scenarios of Figure 2 or 3, assuming TEEX, TEEY and TEEZ are all involved in multi-party secure computation, the first private data can be split into three private shards denoted as U1, U2 and U3.

[83] Step 404: The first provider generates an encryption key corresponding to each private shard.

[84] For example, for the above three private shards U1, U2 and U3, the first provider can generate three random encryption keys: K1, K2 and K3.

[85] Step 406: The first provider uses each encryption key to encrypt the corresponding private shard, and provides the obtained encrypted shards to the trusted computing center.

[86] For example, U1 can be encrypted with K1 to obtain ENC(K1, U1), U2 can be encrypted with K2 to obtain ENC(K2, U2), and U3 can be encrypted with K3 to obtain ENC(K3, U3).

[87] Of course, in actual applications, the first provider can also provide key identifiers

corresponding to each encryption key to the first provider. In one example, the key identifier of any encryption key can be obtained by hash computation, so the key identifier is the hash value of the corresponding encryption key.

[88] Step 408: The trusted computing center stores each encrypted shard in the data storage center.

[89] When the trusted computing center also receives each key identifier, it can store the correspondence between each encrypted shard and key identifier. In one example, the correspondence can be as shown in Table 1.

Table 1

ENC（K1，U1）	K1_ID
ENC（K2，U2）	K2_ID
ENC（K3，U3）	K3_ID

[90] Similarly, other data providers can also upload their respective encrypted shards to the trusted computing center for storage. And each data provider can upload encrypted shards corresponding to multiple data. Therefore, Table 1 can also include identifiers of data providers, identifiers of data, etc. This is not limited herein.

[91] After the first provider uploads encrypted shards of any first private data to the trusted computing center, the first provider can control the computation logic used at the trusted computing center in two ways: first, autonomous control level; second, post hoc accountability level.

[92] The first approach (autonomous control level) is explained first below using the application scenario shown in Figure 2.

[93] In the first approach, the trusted computing center needs to interact with the data provider whenever the computation logic is changed, in order to obtain data authorization. The details are as follows:

[94] Figure 5 shows an interaction diagram of a method for obtaining data authorization according to an embodiment. As shown in Figure 5, the method can comprise at least the following steps:

[95] Step 502: Any first computing node among the plurality of trusted computing nodes receives an authentication request from the first provider.

[96] Here, the first computing node can be a computing module or device with a certain isolation capability to ensure computational security. In one example, the first

computing node is implemented as a trusted enclave. More specifically, the trusted enclave can be implemented using technologies like SGX or TrustZone. In other examples, the first computing node can be implemented in other forms, such as a virtual machine or running instance with security isolation mechanisms.

[97] Taking Figure 2 as an example, the above first computing node can be any node among TEEX, TEEY and TEEZ.

[98] Step 504: Return authentication information to the first provider, wherein the authentication information comprises at least a first code hash of a first computation logic running on the first computing node.

[99] Here, the first computation logic is program code, which can be used for computing average, maximum/minimum values, variance etc. Typically, if any line of code in the first computation logic changes, its corresponding code hash will differ. Thus, the first computation logic can be judged as expected or not based on the first code hash.

[100] In addition, the above authentication information can also include signature information of the first computing node and the public key in the public/private key pair generated by the first computing node, etc. The signature information can be obtained through various signature methods such as hash algorithms, encryption algorithms, etc.

[101] After receiving the above authentication information, the first provider can perform trusted authentication on the first computing node based on the authentication information. Here, the trusted authentication can include verifying the signature information and/or the public key in the public/private key pair, etc.

[102] In addition, the first provider can also verify the correctness of the first code hash. This correctness verification can include comparing the first code hash with a target code hash of a target computation logic that is pre-stored and reviewed by the first provider. If the comparison result matches, the first code hash passes the correctness verification, otherwise it fails.

[103] It should be understood that by comparing the received first code hash with the pre-stored target code hash, the first provider can review the first computation logic in real time, thereby constraining or controlling the computation logic used at the trusted computing center.

[104] Step 506: Receive a channel establishment request sent by the first provider after determining that the first computing node has passed trusted authentication and the first code hash has passed correctness verification.

[105] Step 508: Establish a first trusted channel with the first provider based on the

channel establishment request.

[106] The first trusted channel is also commonly referred to as an encrypted channel, which is typically established based on a regular connection.

[107] Step 510: Obtain, via the first trusted channel, a target encryption key from the respective encryption keys corresponding to the plurality of encrypted shards, thereby obtaining authorization to compute the target encrypted shard corresponding to the target encryption key based on the first computation logic.

[108] Taking Figure 2 as an example, when the first computing node is TEEX, the received target encryption key can be K1; when the first computing node is TEEY, the received target encryption key can be K2.

[109] It should be understood that after receiving the target encryption key, the first computing node can also determine, from the plurality of trusted computing nodes, a second computing node belonging to the same role category as the first computing node, and share the target encryption key with the second computing node, so that the second computing node obtains authorization to compute the target encrypted shard based on the first computation logic. This avoids the problem of the first provider having to send encryption keys to each trusted computing node, thus wasting communication resources.

[110] In addition, after receiving the target encryption key, the first computing node can encrypt the target encryption key using a private key corresponding to the first computing node and the first computation logic. The encrypted target encryption key can be stored in the data storage center. The data storage center can record the correspondence between each trusted computing node and its respectively encrypted encryption key.

[111] The first computing node in the present disclosure provides a Seal capability, where the Seal capability refers to encrypting using private keys of the trusted computing node. Such private keys are inaccessible outside the TEE.

[112] It should be noted that the above private key is usually related to the device environment (e.g. CPU, memory, etc.) and computation logic. That is, different private keys will correspond to different device environments or computation logics. This ensures that when each trusted computing node runs the same computation logic, each trusted computing node can only decrypt the encryption key encrypted by it, and thus only obtain authorization to compute the private shard corresponding to the encryption key. This guarantees data security.

[113] Taking Figure 2 as an example, when TEEX, TEEY and TEEZ all run the same computation logic, the private keys they use to encrypt the respectively received encryption keys K1, K2 and K3 are different, because different trusted computing nodes

usually have different device environments.

[114] Of course, after receiving the target encryption key, the first computing node can also directly keep it in memory instead of storing it to the data storage center.

[115] It should be understood that the above describes the process of each trusted computing node obtaining data authorization.

[116] It should be noted that each time the trusted computing center (i.e. each trusted computing node) changes the computation logic, steps 502-510 are executed repeatedly to obtain data authorization for the current computation logic.

[117] In addition, since the amount of transmission between the data provider and the trusted computing node during the process of obtaining data authorization is independent of the data provider's data size, and is usually a small constant, this solution can help reduce the usage costs for the data provider's data.

[118] It should be understood that after obtaining data authorization, each trusted computing node can jointly compute the respective authorized plaintext shards. The computation process is explained below.

[119] First for the case where the target encryption key is saved to the data storage center:

[120] Taking the above first computing node as an example, the first computing node can receive a computation request from the management center. In response to the computation request, by querying the correspondence between each trusted computing node and its respectively encrypted encryption key, the first computing node can read the encrypted target encryption key from the data storage center, and decrypt it using the decryption key corresponding to the private key, to obtain the target encryption key. Based on the key identifier of the target encryption key, the first computing node can read the corresponding target encrypted shard from the data storage center, decrypt it using the target encryption key to obtain the corresponding target plaintext shard. The first computing node can then compute on the target plaintext shard and other plaintext shards based on the first computation logic.

[121] First, in one example, the first computing node can read, from the data storage center, the target encryption key encrypted by it based on its corresponding node identifier.

[122] Of course, the first computing node may also obtain identifiers of other trusted computing nodes in some way, and thus can obtain the encrypted encryption keys uploaded by other trusted computing nodes. However, since the first computing node cannot know the private keys of other trusted computing nodes, it cannot decrypt them.

That is to say, in this solution, by using its private key to encrypt the corresponding target encryption key, the first computing node can ensure the security of the encryption key.

[123] In addition, the above key identifier of the target encryption key can be computed by the first computing node using a hash algorithm agreed upon with the first provider on the target encryption key.

[124] Next for the case where the target encryption key is directly kept in memory:

[125] The first computing node can receive a computation request from the management center. In response to the computation request, based on the key identifier of the target encryption key, the first computing node can read the corresponding target encrypted shard from the data storage center, decrypt it using the target encryption key to obtain the corresponding target plaintext shard, and compute on the target plaintext shard and other plaintext shards based on the first computation logic.

[126] Taking TEEX in Figure 2 as an example, TEEX can read the encrypted shard ENC(U1, K1) from Table 1 based on the key identifier K1_ID of K1, then decrypt it using K1 to obtain the private shard U1. Similarly, TEEY can decrypt to obtain private shard U2, and TEEZ can decrypt to obtain private shard U3. TEEX, TEEY and TEEZ can then jointly compute on the private shards U1, U2 and U3 based on the first computation logic.

[127] It should be understood that when TEEX, TEEY and TEEZ also obtain encryption keys of private shards from other data providers, the TEEs can also perform computation based on secret sharing.

[128] In summary, in the method for obtaining data authorization provided in the embodiments of the present disclosure, the data provider reviews the computation logic used at the trusted computing center in real time by interacting with the trusted computing center, which constrains the computation logic used at the trusted computing center, thus ensuring the security of the data provider's data.

[129] The second approach (post hoc accountability level) is explained below using the application scenario shown in Figure 3 as an example.

[130] In the second approach, the data provider and trusted computing center can agree on allowed computation logics by contract. So the trusted computing center does not need to interact with the data provider when changing the computation logic, but requires that the computation logic used by the trusted computing center be traceable in real time by the data provider. The details are as follows:

[131] Figure 6 shows an interaction diagram of a method for obtaining data

authorization according to another embodiment. As shown in Figure 6, the method can comprise at least the following steps:

[132] Step 602: Any first proxy node among the proxy nodes obtains a target encryption key from the respective encryption keys corresponding to the plurality of encrypted shards from the first provider.

[133] Optionally, before step 602, the first provider can first perform trusted authentication and correctness verification of code hash on the first proxy node. Here, the code hash corresponding to program code refers to the verification logic running on the first proxy node. The verification logic is used to determine whether a first code hash of a first computation logic running on the corresponding first computing node is verifiably stored.

[134] The specific methods for trusted authentication and correctness verification can refer to steps 502 and 504, and are not repeated here.

[135] After determining that the first proxy node has passed trusted authentication and the verification logic has passed correctness verification, the first provider can establish a trusted channel with the first proxy node and send the target encryption key to the first proxy node through the trusted channel.

[136] Similarly, the first provider can establish trusted channels with other proxy nodes and send encryption keys to other proxy nodes.

[137] Taking Figure 3 as an example, after performing trusted authentication and correctness verification of code hashes on each proxy node TEEA, TEEB and TEEC, the first provider can send the encryption keys K1, K2 and K3 to each proxy node respectively. Thus, when the above first proxy node is TEEA, the received target encryption key is K1; and when the above first proxy node is TEEB, the received target encryption key is K2.

[138] In addition, after receiving the target encryption key, the first proxy node can also encrypt the target encryption key using a private key corresponding to the first proxy node and the verification logic. The encrypted target encryption key can be stored in the data storage center. The data storage center can record the correspondence between each proxy node and its respectively encrypted encryption key.

[139] Of course, after receiving the target encryption key, the first proxy node can also directly keep it in memory instead of storing it to the data storage center.

[140] Step 604: The first proxy node sends an authentication request to the corresponding first computing node.

[141] For example, when the first proxy node is TEEA, it can send an authentication request to the corresponding first computing node TEEX.

[142] Step 606: The first proxy node receives authentication information returned by the first computing node, wherein the authentication information comprises at least a first code hash of a first computation logic running on the first computing node.

[143] Here, the first computation logic is program code, which can be used for computing average, maximum/minimum values, variance etc. Typically, if any line of code in the first computation logic changes, its corresponding code hash will differ. Thus, the first computation logic can be judged as expected or not based on the first code hash.

[144] In addition, the above authentication information can also include signature information of the first computing node and the public key in the public/private key pair generated by the first computing node, etc. The signature information can be obtained through various signature methods such as hash algorithms, encryption algorithms, etc.

[145] Step 608: The first proxy node performs trusted authentication on the first computing node based on the authentication information, and determines whether the first code hash is verifiably stored.

[146] Here, the trusted authentication can include verifying the signature information and/or the public key in the public/private key pair, etc.

[147] The above determination of whether the first code hash is verifiably stored can include obtaining a certificate of verifiable storage that is pre-stored on the first computing node, which includes at least a second code hash of the first computation logic. The first code hash can be compared with the second code hash. If the comparison result matches, it can be determined that the first code hash is verifiably stored.

[148] In one example, the process of the first computing node obtaining the certificate of verifiable storage can be: A trusted third party recognized by the trusted computing center (which can be any device or role) or an administrator provides the second code hash of the first computation logic to the verifiable storage center, and receives the certificate of verifiable storage returned by the verifiable storage center, which contains the second code hash. Afterwards, the trusted third party or administrator recognized by the trusted computing center provides the certificate of verifiable storage to the first computing node for storage.

[149] Step 610: When the trusted authentication passes and the first code hash is verifiably stored, the first proxy node establishes a first trusted channel with the first computing node.

[150] The first trusted channel is also commonly referred to as an encrypted channel, which is typically established based on a regular connection.

[151] Step 612: The first proxy node sends the target encryption key to the first computing node through the first trusted channel, so that the first computing node obtains authorization to compute the target encrypted shard corresponding to the target encryption key based on the first computation logic.

[152] When the target encryption key is stored in the data storage center, the first proxy node can query the correspondence between each proxy node and its respectively encrypted encryption key, and read, from the data storage center, the target encryption key encrypted by it. It can then decrypt it using the decryption key corresponding to the private key to obtain the plaintext target encryption key, and provide the target encryption key to the first computing node.

[153] When the target encryption key is directly kept in memory, the first proxy node can directly read the target encryption key from memory and provide it to the first computing node.

[154] Taking Figure 3 as an example, when the first proxy node is TEEA, it can send the target encryption key K1 to the first computing node TEEX; when the first proxy node is TEEB, it can send the target encryption key K2 to the first computing node TEEY.

[155] Similarly, after receiving the target encryption key, the first computing node can share it with other trusted computing nodes belonging to the same role category as the first computing node.

[156] In addition, the first computing node can keep the target encryption key in memory, or store the target encryption key in the data storage center. The specific storage method refers to the storage method of the first proxy node for the target encryption key, which is not repeated here.

[157] Thus, data authorization for each trusted computing node is completed.

[158] It should be noted that each time the trusted computing center changes the computation logic, steps 604-612 are executed repeatedly to obtain data authorization for the current computation logic.

[159] In addition, in this embodiment, since the data provider distributes the encryption keys to the proxy nodes in advance, the amount of transmission between the proxy nodes and trusted computing nodes during obtaining data authorization is independent of the data provider's data size, and is usually a small constant. Thus, this

solution can help reduce usage costs for the data provider's data.

[160] It should be understood that after obtaining data authorization, each trusted computing node can jointly compute their respective authorized plaintext shards. The computation process is explained below.

[161] Taking the above first computing node as an example, the first computing node can receive a computation request from the management center. In response to the computation request, based on the key identifier of the target encryption key received from the first proxy node, the first computing node can read the corresponding target encrypted shard from the data storage center, decrypt it using the target encryption key to obtain the corresponding target plaintext shard, and compute on the target plaintext shard and other plaintext shards based on the first computation logic. The other plaintext shards can be obtained by other trusted computing nodes decrypting corresponding encrypted shards using their received encryption keys.

[162] It should be understood that the above describes the case where the target encryption key is directly kept in memory. Of course, if the first computing node stores the target encryption key in the data storage center, it would first read and decrypt the encrypted target encryption key from the data storage center, and then read the target encrypted shard based on the key identifier of the decrypted target encryption key.

[163] Taking TEEX in Figure 3 as an example, TEEX can read the encrypted shard ENC(U1, K1) from Table 1 based on the key identifier K1_ID of K1, then decrypt it using K1 to obtain the private shard U1. Similarly, TEEY can decrypt to obtain private shard U2, and TEEZ can decrypt to obtain private shard U3. TEEX, TEEY and TEEZ can then jointly compute on the private shards U1, U2 and U3 based on the first computation logic.

[164] It should be understood that when TEEX, TEEY and TEEZ also obtain encryption keys of private shards from other data providers, the TEEs can also perform computation based on secret sharing.

[165] It should be understood that in the authorization method provided in this embodiment, the first provider did not review the computation logic used by the trusted computing center in real time, but made a determination on whether the code hash is verifiably stored. The review process is performed afterwards, and the specific post hoc process is as follows:

[166] The first provider obtains from the verifiable storage center the second code hash verifiably stored by the first computing node, and sends a data retrieval request to the code storage center, which includes at least the above second code hash. In response to the data retrieval request, the code storage center provides the corresponding first computation logic to the first provider for the first provider to review the legitimacy of

the first computation logic.

[167] In one example, the first provider can perform the above legitimacy review based on the contract it signed with the trusted computing center. The contract is used to record the computation logics jointly agreed upon by both parties.

[168] In summary, in the method for obtaining data authorization provided in the embodiments of the present disclosure, the data provider can review the computation logic used at the trusted computing center afterwards. This allows the trusted computing center to only interact twice when the data provider uploads the data shards and keys, and afterwards when the trusted computing center (i.e. each trusted computing node) changes the computation logic, interaction only needs to occur between the proxy nodes and trusted computing nodes, i.e. communication only occurs internally within the trusted computing center. The internal interaction of the trusted computing center is usually based on high-speed networks, which can greatly improve the efficiency of data authorization and achieve the goal of conserving communication resources.

[169] In addition, in this embodiment, only the proxy nodes can obtain the encryption keys from the data provider. The proxy node will only send the encryption keys to the trusted computing node after verifying the certificate of verifiable storage of the trusted computing node, and the code hash of the trusted computing node is correct. In order for the trusted computing node to obtain the certificate of verifiable storage, it must verifiably store its code hash at the verifiable storage center. Thus, the data provider can obtain code hashes of all trusted computing nodes from the verifiable storage center, and know whether the computation logic matches the original agreement.

[170] Corresponding to the above method for obtaining data authorization, an embodiment of the present disclosure also provides an apparatus for obtaining data authorization, disposed in any first computing node in the trusted computing center. As shown in Figure 7, the apparatus can comprise:

[171] A receiving unit 702, configured to receive an authentication request from the first provider.

[172] A sending unit 704, configured to return authentication information to the first provider, wherein the authentication information comprises at least a first code hash of a first computation logic running on the first computing node.

[173] The receiving unit 702 is further configured to receive a channel establishment request sent by the first provider after determining that the first computing node has passed trusted authentication and the first code hash has passed correctness verification.

[174] Wherein the correctness verification comprises comparing the first code hash with a target code hash of a target computation logic that is pre-stored and reviewed by

the first provider.

[175] An establishment unit 706, configured to establish a first trusted channel with the first provider based on the channel establishment request.

[176] The receiving unit 702 is further configured to obtain, via the first trusted channel, a target encryption key from the respective encryption keys corresponding to the plurality of encrypted shards, thereby obtaining authorization to compute the target encrypted shard corresponding to the target encryption key based on the first computation logic.

[177] Optionally, the apparatus can also comprise:

[178] An encryption unit 708, configured to encrypt the target encryption key using a private key corresponding to the first computing node and the first computation logic.

[179] A storage unit 710, configured to store the encrypted target encryption key in the data storage center.

[180] Optionally, the above trusted computing center also comprises a management center for managing the multiple trusted computing nodes. The above data storage center also maintains respective key identifiers corresponding to the multiple encrypted shards. The apparatus further comprises: a reading unit 712 and a computation unit 714.

[181] The receiving unit 702 is further configured to receive a computation request from the management center.

[182] The reading unit 712 is configured to, in response to the computation request, read the encrypted target encryption key from the data storage center, decrypt it using a decryption key corresponding to the private key to obtain the target encryption key.

[183] The reading unit 712 is further configured to, based on the key identifier of the target encryption key, read the corresponding target encrypted shard from the data storage center, decrypt it using the target encryption key to obtain the corresponding target plaintext shard.

[184] The computation unit 714 is configured to compute on the target plaintext shard and other plaintext shards based on the first computation logic.

[185] Optionally, the apparatus can further comprise:

[186] A determination unit 716, configured to determine, from the multiple trusted computing nodes, a second computing node belonging to the same role category as the first computing node.

[187] A sharing unit 718, configured to share the target encryption key with the second computing node, so that the second computing node obtains authorization to compute the target encrypted shard based on the first computation logic.

[188] The functions of the functional modules of the apparatus in the above embodiments can be implemented through the steps of the method embodiments. Therefore, the specific working process of the apparatus provided in an embodiment is not repeated here.

[189] The apparatus for obtaining data authorization provided in an embodiment can obtain corresponding data authorization from the data provider while saving communication costs.

[190] Corresponding to the above method for obtaining data authorization, an embodiment of the present disclosure also provides a system for obtaining data authorization. As shown in Figure 8, the system comprises a data storage center 802, multiple trusted computing nodes 804, and respective proxy nodes 806. The data storage center 802 maintains at least multiple encrypted shards of the first private data of the first provider.

[191] Each proxy node 806 is configured to obtain from the first provider the encryption keys corresponding to the multiple encrypted shards.

[192] Any first proxy node 806 among the proxy nodes is configured to send an authentication request to the corresponding first computing node 804, and receive authentication information returned by the first computing node 804, wherein the authentication information comprises at least a first code hash of a first computation logic running on the first computing node 804.

[193] The first proxy node 806 is further configured to perform trusted authentication on the first computing node 804 based on the authentication information, and determine whether the first code hash is verifiably stored.

[194] The first proxy node 806 is specifically configured to:

[195] Obtain a certificate of verifiable storage pre-stored on the first computing node 804, wherein the certificate of verifiable storage includes at least a second code hash of the first computation logic;

[196] Compare the first code hash with the second code hash, and if the comparison result matches, determine that the first code hash is verifiably stored.

[197] The certificate of verifiable storage is received and forwarded to the first

computing node 804 by a third party or administrator recognized by the system after the second code hash is provided to the verifiable storage center.

[198] The first proxy node 806 is further configured to establish a first trusted channel with the first computing node 804 when the trusted authentication passes and the first code hash is verifiably stored.

[199] The first proxy node 806 is further configured to send the received target encryption key to the first computing node 804 through the first trusted channel.

[200] The first computing node 804 is configured to obtain authorization to compute the target encrypted shard corresponding to the target encryption key based on the first computation logic.

[201] Optionally, the system further comprises a code storage center, which maintains at least the first computation logic.

[202] The code storage center is configured to receive from the first provider a data retrieval request, which includes at least a second code hash obtained by the first provider from the verifiable storage center.

[203] The code storage center is further configured to provide, in response to the data retrieval request, the first computation logic to the first provider for the first provider to review the legitimacy of the first computation logic.

[204] The functions of the functional modules of the system in the above embodiments can be implemented through the steps of the method embodiments. Therefore, the specific working process of the system provided in an embodiment is not repeated here.

[205] The system for obtaining data authorization provided in an embodiment can obtain corresponding data authorization from the data provider while saving communication costs.

[206] Corresponding to the above method for obtaining data authorization, one embodiment of this specification also provides an apparatus for obtaining data authorization, disposed in any first proxy node in a trusted computing center. The trusted computing center includes a data storage center, multiple trusted computing nodes and corresponding proxy nodes. The data storage center at least maintains multiple encrypted shards of first private data of a first provider. As shown in Figure 9, the apparatus comprises:

[207] An obtaining unit 902, configured to obtain a target encryption key from the respective encryption keys corresponding to the plurality of encrypted shards from the

first provider.

[208] A sending unit 904, configured to send an authentication request to the corresponding first computing node.

[209] A receiving unit 906, configured to receive authentication information returned by the first computing node, wherein the authentication information comprises at least a first code hash of a first computation logic running on the first computing node.

[210] A determination unit 908, configured to perform, based on the authentication information, trusted authentication on the first computing node, and determine whether the first code hash is verifiably stored.

[211] An establishment unit 910, configured to establish a first trusted channel with the first computing node when the trusted authentication passes and the first code hash is verifiably stored.

[212] The sending unit 904 is further configured to send the target encryption key to the first computing node through the first trusted channel, so that the first computing node obtains authorization to compute the target encrypted shard corresponding to the target encryption key based on the first computation logic.

[213] The functions of the functional modules of the apparatus in the above embodiments can be implemented through the steps of the method embodiments. Therefore, the specific working process of the apparatus provided in an embodiment is not repeated here.

[214] The apparatus for obtaining data authorization provided in an embodiment can obtain corresponding data authorization from the data provider while saving communication costs.

[215] According to another aspect, a computer readable storage medium is also provided, storing a computer program which, when executed by a computer, causes the computer to perform the method described in connection with any one of Figures 4 to 6.

[216] According to yet another aspect, a computing device is provided, comprising a memory and a processor, wherein the memory stores executable code which, when executed by the processor, implements the method described in connection with any one of Figures 4 to 6.

[217] The embodiments in the present disclosure are described in a progressive manner. The same or similar parts of the embodiments can refer to each other. Each embodiment focuses on the differences from other embodiments. In particular, for

apparatus embodiments, since they are basically similar to method embodiments, the description is relatively simple, and reference can be made to relevant parts of the method embodiments.

[218] The steps of the methods or algorithms described in combination with the disclosed content of this specification can be implemented in a hardware manner, or can be implemented by a processor executing software instructions. The software instructions can include corresponding software modules, and the software modules can be stored in RAM memory, flash memory, ROM memory, EPROM memory, EEPROM memory, registers, hard disk, mobile hard disk, CD-ROM, or any other form of storage medium known to those skilled in the art. An exemplary storage medium is coupled to a processor to enable the processor to read information from the storage medium and write information to the storage medium. Of course, the storage medium may also be a part of the processor. The processor and storage medium may be located in an ASIC. Additionally, the ASIC may be located in a server. Of course, the processor and storage medium may also exist in the server as separate components.

[219] Those skilled in the art should recognize that in one or more of the above examples, the functions described in the present invention can be implemented using hardware, software, firmware, or any combination thereof. When implemented using software, these functions can be stored on a computer-readable medium or transmitted on a computer-readable medium as one or more instructions or code. Computer readable media includes computer storage media and communication media, where communication media includes any medium that facilitates transfer of computer programs from one place to another. Storage media can be any available media accessible by a general purpose or special purpose computer.

[220] The above describes specific embodiments of this specification. Other embodiments are within the scope of the appended claims. In some cases, the actions or steps recited in the claims may be performed in a different order than in the embodiments and still achieve the desired results. Additionally, the processes depicted in the figures do not necessarily require the specific order or sequential order shown in order to achieve desirable results. In certain embodiments, multitasking and parallel processing may be possible or advantageous.

[221] The above specific implementation manners further detail the objectives, technical solutions, and beneficial effects of this specification, and it should be understood that the above descriptions are merely specific implementation manners of this specification and are not intended to limit the protection scope of this specification. Any modifications, equivalent replacements, improvements, etc. made on the basis of the technical solutions of this specification shall fall within the protection scope of this specification.

Claims

1. A method for obtaining data authorization, performed by a trusted computing center; the trusted computing center comprises a data storage center and multiple trusted computing nodes, the data storage center maintains at least multiple encrypted shards of first private data of a first provider; the method comprises:

receiving, by any first computing node among the multiple trusted computing nodes, an authentication request from the first provider;

Returning authentication information to the first provider, wherein the authentication information comprises at least a first code hash of a first computation logic running on the first computing node;

Receiving a channel establishment request sent by the first provider after determining that the first computing node has passed trusted authentication and the first code hash has passed correctness verification;

Establishing a first trusted channel with the first provider based on the channel establishment request; and

Obtaining, via the first trusted channel, a target encryption key from the respective encryption keys corresponding to the multiple encrypted shards, thereby obtaining authorization to compute the target encrypted shard corresponding to the target encryption key based on the first computation logic.

2. The method according to claim 1, further comprising:

encrypting, by the first computing node, the target encryption key using a private key corresponding to the first computing node and the first computation logic;

Storing the encrypted target encryption key in the data storage center.

3. The method according to claim 2, wherein the trusted computing center further comprises a management center for managing the multiple trusted computing nodes; the data storage center also maintains respective key identifiers corresponding to the multiple encrypted shards; the method further comprises:

receiving, by the first computing node, a computation request from the management center;

In response to the computation request, reading the encrypted target encryption key from the data storage center, decrypting it using a decryption key corresponding to the private key to obtain the target encryption key;

Based on the key identifier of the target encryption key, reading a corresponding target encrypted shard from the data storage center, decrypting it using the target encryption key to obtain a corresponding target plaintext shard; and

Computing on the target plaintext shard and other plaintext shards based on the first computation logic.

4. The method according to claim 1, further comprising:

determining, by the first computing node from the multiple trusted computing nodes, a second computing node belonging to the same role category as the first computing node;

Sharing the target encryption key with the second computing node, so that the second computing node obtains authorization to compute the target encrypted shard based on the first computation logic.

5. The method according to claim 1, wherein the correctness verification comprises comparing the first code hash with a target code hash of a target computation logic that is pre-stored and reviewed by the first provider.

6. A method for obtaining data authorization, performed by a trusted computing center; the trusted computing center comprises a data storage center, multiple trusted computing nodes, and respective proxy nodes; the data storage center maintains at least multiple encrypted shards of first private data of a first provider; the method comprises:

obtaining, by each proxy node from the first provider, the encryption keys corresponding to the multiple encrypted shards;

sending, by any first proxy node among the proxy nodes, an authentication request to a corresponding first computing node, and receiving authentication information returned by the first computing node, wherein the authentication information comprises at least a first code hash of a first computation logic running on the first computing node;

performing, by the first proxy node based on the authentication information, trusted authentication on the first computing node, and determining whether the first code hash is verifiably stored;

When the trusted authentication passes and the first code hash is verifiably stored, establishing, by the first proxy node, a first trusted channel with the first computing node;

sending, by the first proxy node, the received target encryption key to the first computing node through the first trusted channel; and

obtaining, by the first computing node, authorization to compute the target encrypted shard corresponding to the target encryption key based on the first computation logic.

7. The method according to claim 6, wherein determining whether the first code hash is verifiably stored comprises:

Obtaining from the first computing node a certificate of verifiable storage pre-stored on it, wherein the certificate of verifiable storage includes at least a second code hash of the first computation logic;

Comparing the first code hash with the second code hash; if the comparison result matches, determining that the first code hash is verifiably stored.

8. The method according to claim 7, wherein the certificate of verifiable storage is received and forwarded to the first computing node by a third party or administrator recognized by the trusted computing center after the second code hash is provided to a verifiable storage center.

9. The method according to claim 8, wherein the trusted computing center further comprises a code storage center, which maintains at least the first computation logic; the method further comprises:

receiving, by the code storage center, a data retrieval request from the first provider, which includes at least the second code hash obtained by the first provider from the verifiable storage center;

In response to the data retrieval request, providing, by the code storage center, the first computation logic to the first provider for the first provider to review the legitimacy of the first computation logic.

10. A method for obtaining data authorization, involving a trusted computing center; the trusted computing center comprises a data storage center, multiple trusted computing nodes, and respective proxy nodes corresponding to the trusted computing nodes; the data storage center maintains at least multiple encrypted shards of first private data of a first provider; the method is performed by any first proxy node among the proxy nodes, comprising:

Obtaining from the first provider a target encryption key from the respective encryption keys corresponding to the multiple encrypted shards;

Sending an authentication request to a corresponding first computing node;

Receiving authentication information returned by the first computing node, wherein the

authentication information comprises at least a first code hash of a first computation logic running on the first computing node;

Performing, based on the authentication information, trusted authentication on the first computing node, and determining whether the first code hash is verifiably stored;

When the trusted authentication passes and the first code hash is verifiably stored, establishing a first trusted channel with the first computing node; and

Sending the target encryption key to the first computing node through the first trusted channel; thereby the first computing node obtains authorization to compute the target encrypted shard corresponding to the target encryption key based on the first computation logic.

11. An apparatus for obtaining data authorization, disposed in any first computing node of a trusted computing center; the trusted computing center comprises a data storage center and multiple trusted computing nodes, the data storage center maintains at least multiple encrypted shards of first private data of a first provider; the apparatus comprises:

a receiving unit, configured to receive an authentication request from the first provider;

a sending unit, configured to return authentication information to the first provider, wherein the authentication information at least comprises a first code hash of a first computing logic running in the first computing node;

the receiving unit is further configured to receive a channel establishment request sent by the first provider after determining that the first computing node passes trusted authentication and the first code hash passes correctness verification;

an establishment unit, configured to establish a first trusted channel with the first provider according to the channel establishment request; and

the receiving unit is further configured to receive, through the first trusted channel, target encryption keys corresponding to the multiple encrypted shards to obtain permission to perform computation on a target encrypted shard corresponding to the target encryption key based on the first computing logic.

12. The apparatus according to claim 11, further comprising:

an encryption unit, configured to encrypt the target encryption key using a private key corresponding to the first computing node and the first computing logic;

a storage unit, configured to store the encrypted target encryption key to the data storage

center.

13. The apparatus according to claim 12, wherein the trusted computing center further comprises a management center for managing the multiple trusted computing nodes; the data storage center also maintains respective key identities corresponding to the multiple encrypted shards; the apparatus further comprises: a reading unit and a computing unit;

the receiving unit is further configured to receive a computation request from the management center;

the reading unit is configured to, in response to the computation request, read the encrypted target encryption key from the data storage center, decrypt it using a decryption key corresponding to the private key to obtain the target encryption key;

the reading unit is further configured to, based on the key identity of the target encryption key, read a corresponding target encrypted shard from the data storage center, decrypt it using the target encryption key to obtain a corresponding target plaintext shard; and

the computing unit is configured to perform computation on the target plaintext shard and other plaintext shards based on the first computing logic.

14. The apparatus according to claim 11, further comprising:

a determining unit, configured to determine, from the multiple trusted computing nodes, a second computing node that belongs to the same role category as the first computing node;

a sharing unit, configured to share the target encryption key with the second computing node, so that the second computing node obtains permission to perform computation on the target encrypted shard based on the first computing logic.

15. The apparatus according to claim 11, wherein the correctness verification comprises comparing the first code hash with a target code hash of a target computing logic that is pre-stored and reviewed by the first provider.

16. A system for obtaining data authorization, comprising a data storage center, multiple trusted computing nodes, and corresponding proxy nodes; the data storage center maintains at least multiple encrypted shards of first private data of a first provider;

the proxy nodes are configured to obtain, from the first provider, encryption keys corresponding to the multiple encrypted shards;

any first proxy node among the proxy nodes is configured to send an authentication request to a corresponding first computing node, and receive authentication information returned by the first computing node, wherein the authentication information at least comprises a first code hash of a first computing logic running in the first computing node;

the first proxy node is further configured to perform trusted authentication on the first computing node based on the authentication information, and determine whether the first code hash is verifiably stored;

the first proxy node is further configured to, when the trusted authentication passes and the first code hash is verifiably stored, establish a first trusted channel with the first computing node;

the first proxy node is further configured to send a target encryption key received to the first computing node through the first trusted channel; and

the first computing node is configured to obtain permission to perform computation on a target encrypted shard corresponding to the target encryption key based on the first computing logic.

17. The system according to claim 16, wherein the first proxy node is specifically configured to:

obtain a certificate of verifiable storage stored in advance in the first computing node, wherein the certificate of verifiable storage at least comprises a second code hash of the first computing logic;

compare the first code hash with the second code hash, and if the comparison result is consistent, determine that the first code hash is verifiably stored.

18. The system according to claim 17, wherein the certificate of verifiable storage is received and forwarded to the first computing node by the trusted third party or administrator after the trusted third party or administrator provides the second code hash to a verifiable storage center.

19. The system according to claim 18, wherein the system further comprises a code storage center that maintains at least the first computing logic;

the code storage center is configured to receive a data acquisition request from the first provider, wherein the data acquisition request at least comprises the second code hash acquired by the first provider from the verifiable storage center;

the code storage center is further configured to provide the first computing logic to the

first provider in response to the data acquisition request for the first provider to review the legitimacy of the first computing logic.

20. An apparatus for obtaining data authorization, disposed in any first proxy node of a trusted computing center; the trusted computing center comprises a data storage center, multiple trusted computing nodes, and corresponding proxy nodes; the data storage center maintains at least multiple encrypted shards of first private data of a first provider; the apparatus comprises:

an obtaining unit, configured to obtain, from the first provider, a target encryption key among encryption keys corresponding to the multiple encrypted shards;

a sending unit, configured to send an authentication request to a corresponding first computing node;

a receiving unit, configured to receive authentication information returned by the first computing node, wherein the authentication information at least comprises a first code hash of a first computing logic running in the first computing node;

a determining unit, configured to perform trusted authentication on the first computing node based on the authentication information, and determine whether the first code hash is verifiably stored;

an establishment unit, configured to, when the trusted authentication passes and the first code hash is verifiably stored, establish a first trusted channel with the first computing node;

the sending unit is further configured to send the target encryption key to the first computing node through the first trusted channel; so that the first computing node obtains permission to perform computation on a target encrypted shard corresponding to the target encryption key based on the first computing logic.

21. A computer readable storage medium, storing a computer program, wherein when the computer program is executed in a computer, the computer program causes the computer to perform the method according to any one of claims 1-10.

22. A computing device comprising a memory and a processor, wherein the memory stores executable code which when executed by the processor causes the computing device to perform the method according to any one of claims 1-10.