

MULTIPARTY SECURE COMPUTING METHOD, DEVICE, AND ELECTRONIC DEVICE

TECHNICAL FIELD

[0001] One or more embodiments of this specification are related to the field of distributed computing, and in particular, to a secure multi-party computation method and apparatus, and an electronic device.

BACKGROUND

[0002] Secure multi-party computation is a cryptography technology. A plurality of parties participating in computation may jointly perform secure computation based on respectively held inputted data by executing a secure multi-party computation algorithm, to obtain a computation result without leaking the held inputted data to other parties.

[0003] Although the secure multi-party computation algorithm can protect private data, an incorrect or even malicious algorithm implementation is still prone to leakage of private data. In an ideal scenario, the plurality of parties may develop respective secure multi-party computation software and respectively use the secure multi-party computation software developed respectively to participate in the secure multi-party computation. However, in actual applications, secure multi-party computation software usually used by the plurality of parties may be provided by one provider, so that the provider needs to prove that the provided secure multi-party computation software is trustworthy and there is no leakage of private data.

SUMMARY

[0004] This specification provides a secure multi-party computation method, applicable to any computing node deployed in a distributed network, a plurality of computing nodes being deployed in the distributed network, the plurality of computing nodes jointly participating in a secure multi-party computation based on respectively held private data, the computing node being connected to a trusted key source, the method including:

[0005] obtaining a trusted key from the trusted key source;

[0006] encrypting the held private data based on the obtained trusted key to obtain ciphertext data; and

[0007] transmitting at least the ciphertext data as a computing parameter to other computing

nodes participating in the secure multi-party computation for the other computing nodes to perform the secure multi-party computation based on collected computing parameters transmitted by the computing nodes participating in the secure multi-party computation.

[0008] Optionally, the method further includes:

[0009] creating an audit log corresponding to the ciphertext data transmitted by the computing node to the other computing nodes, the audit log recording description information related to the ciphertext data.

[0010] Optionally, the method further includes:

[0011] collecting computing parameters transmitted by the other computing nodes participating in the secure multi-party computation, the computing parameters including at least ciphertext data obtained by encrypting the private data held by the other computing nodes;

[0012] performing the secure multi-party computation based on the collected computing parameters; and

[0013] creating an audit log corresponding to the ciphertext data transmitted by the other computing nodes, the audit log including description information related to the encrypted private data.

[0014] Optionally, the method further includes:

[0015] performing, based on the audit log, a privacy audit on the ciphertext data transmitted by the computing node participating in the secure multi-party computation, to determine whether the ciphertext data transmitted as a computing parameter by the computing node contains a risk of privacy leakage; or

[0016] transmitting the audit log to a third-party audit node for the audit node to perform a privacy audit on the ciphertext data transmitted as a computing parameter by the computing node participating in the secure multi-party computation, to determine whether the ciphertext data transmitted as a computing parameter by the computing node contains a risk of privacy leakage.

[0017] Optionally, the description information related to the ciphertext data includes an encryption key corresponding to the ciphertext data; and

[0018] the performing, based on the audit log, a privacy audit on the ciphertext data transmitted by the computing node participating in the secure multi-party computation, to determine whether the ciphertext data transmitted as a computing parameter by the computing node contains a risk of privacy leakage includes:

[0019] verifying whether the encryption key that is recorded in the audit log and corresponds to the ciphertext data transmitted by the computing node participating in the secure multi-party

computation matches the trusted key from the trusted key source; and

[0020] if it does, determining that the ciphertext data transmitted as the computing parameter by the computing node does not contain the risk of privacy leakage; or

[0021] if it does not, determining that the ciphertext data transmitted as the computing parameter by the computing node contains the risk of privacy leakage.

[0022] Optionally, a decryption key corresponding to the ciphertext data and cleartext data corresponding to the ciphertext data are further recorded in the audit log corresponding to the ciphertext data transmitted by the computing node to the other computing nodes; and

[0023] the method further includes:

[0024] decrypting, based on the decryption key that is recorded in the audit log and corresponds to the ciphertext data transmitted by the computing node participating in the secure multi-party computation, the ciphertext data to obtain cleartext data;

[0025] verifying whether the decrypted cleartext data matches the cleartext data that is recorded in the audit log and corresponds to the ciphertext data; and

[0026] if it does, determining that the ciphertext data transmitted as the computing parameter by the computing node does not contain the risk of privacy leakage; or

[0027] if it does not, determining that the ciphertext data transmitted as the computing parameter by the computing node contains the risk of privacy leakage.

[0028] Optionally, the trusted key is a random number; and

[0029] the obtaining a trusted key from the trusted key source includes any one of the following:

[0030] obtaining a random number generated by the trusted key source as the trusted key;

[0031] obtaining a random number generated by the trusted key source, and using a result obtained by further performing an operation on the random number generated by the trusted key source and a random number generated by the computing node as the trusted key;

[0032] obtaining a random number generated by the trusted key source, and using a random number calculated by inputting the random number into a key generation function as the trusted key; and

[0033] obtaining a random number generated by the trusted key source, and using a result obtained by further performing an operation on a random number calculated by inputting the random number into a key generation function and a random number generated by the computing node as the trusted key.

[0034] This specification further provides a secure multi-party computation apparatus, applicable to any computing node deployed in a distributed network, a plurality of computing

nodes being deployed in the distributed network, the plurality of computing nodes jointly participating in a secure multi-party computation based on respectively held private data, the computing node being connected to a trusted key source, the method including:

[0035] an obtaining module, configured to obtain a trusted key from the trusted key source;

[0036] an encryption module, configured to encrypt the held private data based on the obtained trusted key to obtain ciphertext data; and

[0037] a transmission module, configured to transmit at least the ciphertext data as a computing parameter to other computing nodes participating in the secure multi-party computation for the other computing nodes to perform the secure multi-party computation based on collected computing parameters transmitted by the computing nodes participating in the secure multi-party computation.

[0038] Optionally, the apparatus further includes:

[0039] a creation module, configured to create an audit log corresponding to the ciphertext data transmitted by the computing node to the other computing nodes, the audit log recording description information related to the ciphertext data.

[0040] Optionally, the apparatus further includes:

[0041] a collection module, configured to collect computing parameters transmitted by the other computing nodes participating in the secure multi-party computation, the computing parameters including at least ciphertext data obtained by encrypting the private data by held the other computing nodes; and

[0042] a computation module, configured to perform the secure multi-party computation based on the collected computing parameters, where

[0043] the creation module is further configured to:

[0044] create an audit log corresponding to the ciphertext data transmitted by the other computing nodes, the audit log including description information related to the encrypted private data.

[0045] Optionally, the apparatus further includes:

[0046] an audit module, configured to perform, based on the audit log, a privacy audit on the ciphertext data transmitted by the computing node participating in the secure multi-party computation, to determine whether the ciphertext data transmitted as a computing parameter by the computing node contains a risk of privacy leakage; or

[0047] transmit the audit log to a third-party audit node for the audit node to perform a privacy audit on the ciphertext data transmitted as a computing parameter by the computing node participating in the secure multi-party computation, to determine whether the ciphertext

data transmitted as a computing parameter by the computing node contains a risk of privacy leakage.

[0048] Optionally, the description information related to the ciphertext data includes an encryption key corresponding to the ciphertext data; and

[0049] the audit module is further configured to:

[0050] verify whether the encryption key that is recorded in the audit log and corresponds to the ciphertext data transmitted by the computing node participating in the secure multi-party computation matches the trusted key from the trusted key source; and

[0051] if it does, determine that the ciphertext data transmitted as the computing parameter by the computing node does not contain the risk of privacy leakage; or

[0052] if it does not, determine that the ciphertext data transmitted as the computing parameter by the computing node contains the risk of privacy leakage.

[0053] Optionally, a decryption key corresponding to the ciphertext data and cleartext data corresponding to the ciphertext data are further recorded in the audit log corresponding to the ciphertext data transmitted by the computing node to the other computing nodes; and

[0054] the audit module is further configured to:

[0055] decrypt, based on the decryption key that is recorded in the audit log and corresponds to the ciphertext data transmitted by the computing node participating in the secure multi-party computation, the ciphertext data to obtain cleartext data;

[0056] verify whether the decrypted cleartext data matches the cleartext data that is recorded in the audit log and corresponds to the ciphertext data; and

[0057] if it does, determine that the ciphertext data transmitted as the computing parameter by the computing node does not contain the risk of privacy leakage; or

[0058] if it does not, determine that the ciphertext data transmitted as the computing parameter by the computing node contains the risk of privacy leakage.

[0059] Optionally, the trusted key is a random number; and

[0060] obtaining a random number generated by the trusted key source as the trusted key;

[0061] obtaining a random number generated by the trusted key source, and using a result obtained by further performing an operation on the random number generated by the trusted key source and a random number generated by the computing node as the trusted key;

[0062] obtaining a random number generated by the trusted key source, and using a random number calculated by inputting the random number into a key generation function as the trusted key; and

[0063] obtaining a random number generated by the trusted key source, and using a result

obtained by further performing an operation on a random number calculated by inputting the random number into a key generation function and a random number generated by the computing node as the trusted key.

[0064] This specification further provides an electronic device, including:

[0065] a processor; and

[0066] a memory configured to store machine-executable instructions, where

[0067] by reading and executing the machine-executable instructions stored in the memory and corresponding to control logic of secure multi-party computation, the processor is configured to:

[0068] obtain a trusted key from the trusted key source;

[0069] encrypt the held private data based on the obtained trusted key to obtain ciphertext data; and

[0070] transmit at least the ciphertext data as a computing parameter to other computing nodes participating in the secure multi-party computation for the other computing nodes to perform the secure multi-party computation based on collected computing parameters transmitted by the computing nodes participating in the secure multi-party computation.

[0071] According to the foregoing technical solution, the computing nodes participating in the secure multi-party computation may obtain a trusted key from the connected trusted key source, encrypt the held private data based on the obtained trusted key to obtain ciphertext data, and then transmit at least the ciphertext data as a computing parameter to other computing nodes participating in the secure multi-party computation to perform the secure multi-party computation. In this way, while performing encrypted computation on the held private data, the computing node may no longer use a key generated locally by using a carried secure multi-party computation algorithm, and use the trusted key obtained from the connected trusted key source instead, so that the risk of privacy leakage caused by constructing a key according to a malicious secure multi-party computation algorithm based on the held private data to encrypt the private data can be avoided.

BRIEF DESCRIPTION OF THE DRAWINGS

[0072] FIG. 1 is a flowchart of a secure multi-party computation method according to an exemplary embodiment.

[0073] FIG. 2 is a schematic structural diagram of an electronic device according to an exemplary embodiment.

[0074] FIG. 3 is a block diagram of a secure multi-party computation apparatus according to an exemplary embodiment.

DETAILED DESCRIPTION OF THE INVENTION

[0075] For a computing node carrying secure multi-party computation algorithms such as a homomorphic encryption algorithm and a garbled circuit algorithm, one or more random numbers are generally generated as random keys based on carried randomized algorithms, the generated random key is used to perform encrypted computation on held private data to obtain ciphertext data, and the computed ciphertext data is then transmitted as a computing parameter to other computing nodes participating in a secure multi-party computation.

[0076] However, in actual applications, in a malicious secure multi-party computation algorithm, a known key held by other computing nodes participating in the secure multi-party computation may be used to encrypt the held private data. In this way, when the ciphertext data is transmitted to the other computing nodes participating in the secure multi-party computation, the other computing nodes may decrypt the ciphertext data based on the held known key to obtain the private data held by other computing nodes, leading to privacy leakage.

[0077] Based on this, this specification is intended to provide a technical solution of encrypting held private data based on a trusted key from a trusted key source, to avoid privacy leakage caused by encrypting the private data based on a known key held by a computing node participating in the secure multi-party computation.

[0078] During implementation, computing nodes in a distributed network may further be connected to a trusted key source respectively in addition to carrying a secure multi-party computation algorithm provided by a provider, where the foregoing trusted key sources may be key sources independently developed by operators of the computing nodes or may be key sources developed by a trusted third party other than the foregoing provider.

[0079] When the computing node participates in a secure multi-party computation based on held private data, the computing node may obtain a trusted key from the connected trusted key source, encrypt the held private data based on the trusted key to obtain ciphertext data, and then transmit at least the ciphertext data as a computing parameter to other computing nodes participating in the secure multi-party computation in the distributed network, so that the other computing nodes perform the secure multi-party computation based on collected computing parameters transmitted by the computing nodes participating in the secure multi-party computation.

[0080] According to the foregoing technical solution, the computing nodes participating in the secure multi-party computation may obtain a trusted key from the connected trusted key source, encrypt the held private data based on the obtained trusted key to obtain ciphertext data, and then transmit at least the ciphertext data as a computing parameter to other computing nodes participating in the secure multi-party computation to perform the secure multi-party computation. In this way, while performing encrypted computation on the held private data, the computing node may no longer use a key generated locally by using a carried secure multi-party computation algorithm, and use the trusted key obtained from the connected trusted key source instead, so that the risk of privacy leakage caused by constructing a key according to a malicious secure multi-party computation algorithm based on the held private data to encrypt the private data can be avoided.

[0081] This specification is described below with reference to specific embodiments and specific application scenarios.

[0082] FIG. 1 shows a secure multi-party computation method according to an embodiment of this specification, applicable to any computing node deployed in a distributed network, a plurality of computing nodes being deployed in the distributed network, the plurality of computing nodes jointly participating in a secure multi-party computation based on respectively held private data, where the computing nodes deployed in the distributed network are respectively connected to a trusted key source, and the secure multi-party computation method performs the following steps:

[0083] Step 102: Obtain a trusted key from the trusted key source.

[0084] Step 104: Encrypt the held private data based on the obtained trusted key to obtain ciphertext data.

[0085] Step 106: Transmit at least the ciphertext data as a computing parameter to other computing nodes participating in the secure multi-party computation for the other computing nodes to perform the secure multi-party computation based on collected computing parameters transmitted by the computing nodes participating in the secure multi-party computation.

[0086] The distributed network may include a data network in any form that can support secure multi-party computation and uses a distributed architecture.

[0087] For example, in one scenario, the distributed network may be a secure multi-party computation platform constructed by a device cluster (for example, a server cluster) formed by a plurality of computing devices.

[0088] In another example, in another scenario, the distributed network may be alternatively a block chain network formed by a plurality of node devices, where an online service in which

the plurality of node devices need to jointly participate in a secure multi-party computation based on held private data and it needs to be ensured that the held private data is not disclosed to other node devices may be deployed in the block chain network. A specific type of the online service is not specifically limited in this specification.

[0089] The computing node may include a node device that can carry the secure multi-party computation algorithm in the distributed network.

[0090] The secure multi-party computation algorithm may include secure multi-party computation algorithms that are independently developed by the computing nodes or centrally provided by a provider. For example, in actual applications, the secure multi-party computation algorithm may, for example, correspond to secure multi-party computation software independently developed by the computing nodes or centrally provided by a provider.

[0091] The computing parameters are, for example, computing parameters generated by the computing nodes participating in the secure multi-party computation based on the carried secure multi-party computation algorithms and related to the private data held by the computing nodes; and the computing nodes need to transmit generated computing parameters to each other through an exchange message, and perform the secure multi-party computation based on the collected computing parameters generated by the computing nodes.

[0092] In actual applications, the computing parameters may generally include random numbers needing to participate in a secure multi-party computation; and/or ciphertext data needing to participate in a secure multi-party computation.

[0093] It should be noted that, in actual applications, the foregoing secure multi-party computation algorithm generally includes secure multi-party computation algorithms such as a secret sharing algorithm, a homomorphic encryption algorithm, a garbled circuit algorithm, and the like. Specific content included by the computing parameters generally depends on specific algorithm types of the secure multi-party computation algorithms carried by the computing nodes. When the node devices carry the different algorithms listed above, the content included by the computing parameters may be somewhat different.

[0094] For example, based on the secret sharing algorithm, the computing parameter generally includes N random numbers obtained by splitting the held private data by the computing nodes, where the N random numbers include N-1 independently generated random numbers and an operation result (that is, an Nth random number) obtained by performing an operation on the N-1 random numbers and the held private data.

[0095] Based on the homomorphic encryption algorithm, the computing parameter generally includes ciphertext data obtained by encrypting the private data based on a held key.

[0096] Based on the garbled circuit algorithm, the computing parameter generally includes random numbers (which may also be referred to as salt values of an input value and an output value) used for respectively replacing an input value (that is, inputted data participating in computation) and an output value of a garbled circuit, and ciphertext data obtained by encrypting the salt value of outputted data by using the salt value of the input value as a key.

[0097] The trusted key source may include key sources independently developed by operators of the computing nodes, or may include key sources provided by a trusted third party other than the foregoing provider. For example, in actual applications, the trusted key source may be, for example, a computer program (for example, a trusted key generation algorithm or a trusted key generation function) independently developed by the operators of the computing nodes and used for generating a trusted key source, or may be a computer program developed by a trusted third party other than the provider and used for generating a trusted key source.

[0098] In this specification, a plurality of computing nodes may be deployed in a distributed network, as participants of the secure multi-party computation, the computing nodes may jointly carry the same type of secure multi-party computation algorithm to participate in the secure multi-party computation without leaking the respectively held private data.

[0099] For example, the computing nodes may jointly carry secure multi-party computation software provided by the provider of the secure multi-party computation algorithm and perform message exchange by using the secure multi-party computation software to transmit ciphertext data participating in the secure multi-party computation between the computing nodes, to participate in the secure multi-party computation.

[0100] It should be noted that, in this specification, the secure multi-party computation algorithm carried in the computing nodes is, for example, an algorithm that supports transmission of the ciphertext data obtained by encrypting the held private data based on a key as a computing parameter to other computing nodes participating in the secure multi-party computation. The secure multi-party computation algorithm is, for example, a homomorphic encryption algorithm or a garbled circuit algorithm.

[0101] In addition to jointly carrying the same type of secure multi-party computation algorithm, an independent trusted key source, an independent data source, a result storage system, and a log storage system may further be separately deployed in the computing nodes deployed in the distributed network.

[0102] The trusted key source is used for generating a random trusted key.

[0103] The data source is used for storing and maintaining private data held by data nodes.

[0104] The result storage system is used for storing and maintaining a computation result

eventually obtained by participating in a secure multi-party computation by the data nodes.

[0105] It should be noted that, in actual applications, the trusted key source may be alternatively, for example, a key source provided by a trusted third party other than the provider of the secure multi-party computation algorithm. In this case, alternatively, a trusted key source may not be deployed locally in the computing nodes, but instead a third-party trusted key source is directly connected to the computing nodes, to obtain a trusted key source used for encrypting the held private data.

[0106] In addition, in some secure multi-party computation in which only a small amount of data participates, it may be alternatively not necessary to deploy an independent data source and a result storage system in the node devices. In this case, the computing nodes may directly obtain data inputted by a user through an operation interface as the data source, and output a final computation result of the secure multi-party computation to the user through the operation interface.

[0107] The log storage system is used for storing and maintaining audit logs created by the data nodes and related to the computing parameters.

[0108] It should be noted that, the created audit logs may be alternatively stored in a third-party storage device or storage system. In this case, it may not be necessary to deploy an independent log storage system locally in the computing nodes. That is, in actual applications, a generation node and a storage node of an audit log may be the same or different, and this is not specifically limited in this specification.

[0109] For any target computing node participating in the secure multi-party computation, when jointly completing the secure multi-party computation with other computing nodes, the target computing node may first read private data needing to participate in the secure multi-party computation from deployed data sources, and obtain a trusted key from the trusted key source.

[0110] The target computing node may then perform encrypted computation on the read private data based on the obtained trusted key, to generate ciphertext data corresponding to the held private data.

[0111] It should be noted that the trusted key generated by the trusted key source is generally a random number generated based on a key generation function.

[0112] In a shown implementation, during the obtaining of a trusted key from a trusted key source, the target computing node may directly obtain a random number generated by the trusted key source as the trusted key; or the target computing node may only obtain a random number (that is, a random number seed used for generating a random number) used for

generating a random number from the trusted key source, perform computation locally by inputting the random number into a key generation function, and then use a computed random number as the trusted key.

[0113] In another shown implementation, during the obtaining of a trusted key from a trusted key source, the target computing node may alternatively use a result obtained by further performing an operation on a random number obtained from the trusted key source and a random number generated locally based on a carried key generation function as the trusted key; or the target computing node may alternatively use a result obtained by further performing an operation on a random number generated by inputting a random number obtained from the trusted key source into a key generation function for computation and a random number generated locally based on a carried random number generation function (that is, using a random number seed generated locally to compute a random number) as the trusted key.

[0114] After the target computing node performs encrypted computation on the read private data based on the obtained trusted key to generate ciphertext data corresponding to the held private data, the target computing node may construct an exchange message based on the carried secure multi-party computation algorithm and an existing message exchange mechanism, to transmit at least the generated ciphertext data as a computing parameter to other computing nodes participating in the secure multi-party computation.

[0115] The other computing nodes may collect computing parameters transmitted, through the exchange messages, by the computing nodes participating in the secure multi-party computation, perform the secure multi-party computation based on the collected computing parameters, and store the computation result in the result storage system deployed locally.

[0116] Correspondingly, the target computing node may alternatively collect computing parameters transmitted, through the exchange messages, by the computing nodes participating in the secure multi-party computation, perform the secure multi-party computation based on the collected computing parameters, and store the computation result in the result storage system deployed locally.

[0117] It should be noted that a computation process in which the computing nodes participating in the secure multi-party computation perform the secure multi-party computation based on the collected computing parameters generally depends on a specific algorithm type of the secure multi-party computation algorithm carried in the computing nodes. Generally, computation processes involved in different types of secure multi-party computation algorithms are somewhat different.

[0118] A brief description of algorithm logic of various algorithms listed above is made

below by using an example in which the secure multi-party computation algorithms carried in the computing nodes are the homomorphic encryption algorithm and the garbled circuit algorithm.

[0119] 1) Homomorphic encryption algorithm

[0120] The homomorphic encryption algorithm is a type of encryption algorithm that can perform data operation in a ciphertext state. A ciphertext of the computation result obtained by performing computation based on a cleartext may also be obtained by performing computation on the ciphertext without decryption and the use of a decryption key.

[0121] For example, an operation is performed on a homomorphic ciphertext $E(a)$ of a and a homomorphic ciphertext $E(b)$ of b to obtain a ciphertext $E(a+b)$ of $a+b$, a , b or $a+b$ is not leaked in the operation process, and a decryption key does not need to be used for decrypting the ciphertext.

[0122] If the secure multi-party computation algorithm carried in the computing nodes deployed in the distributed network is the homomorphic encryption algorithm, after the target computing node reads private data S needing to participate in the secure multi-party computation from a deployed data source, based on existing computation logic of the homomorphic encryption algorithm, a pair of encryption key PK and decryption key SK for homomorphic encryption are generally generated based on the homomorphic encryption algorithm.

[0123] In this specification, the encryption key PK and the decryption key SK may no longer be generated according to the homomorphic encryption algorithm, but instead are generated by a trusted key source deployed locally in the computing node.

[0124] In this case, the target computing node may obtain the encryption key PK and the decryption key SK as trusted keys from a trusted key source deployed locally, encrypt the private data based on the encryption key PK to generate ciphertext data, then construct an exchange message based on an existing message exchange mechanism of the homomorphic encryption algorithm, and respectively transmit the generated ciphertext data to other computing nodes participating in the secure multi-party computation.

[0125] According to one aspect, the other computing nodes may also use the same mode, to obtain a pair of encryption key PK and decryption key SK for homomorphic encryption as trusted keys from a deployed trusted key source, encrypt private data read from a locally deployed data source based on the encryption key PK to obtain ciphertext data, collect ciphertext data transmitted by the computing nodes and obtained by encrypting the private data held by the computing nodes, and perform computation on collected ciphertext data transmitted

by the other computing nodes and the ciphertext data generated locally together according to an appointed operation logic, to obtain ciphertext data corresponding to a final computation result obtained by performing a secure multi-party computation on the private data held by the computing nodes (the ciphertext data cannot be decrypted locally by the other computing nodes).

[0126] After the ciphertext data corresponding to the final computation result is obtained, the other computing nodes may broadcast the ciphertext data corresponding to the final computation result to node devices. After the target computing node receives the ciphertext data corresponding to the final computation result, the target computing node may decrypt the received ciphertext data corresponding to the final computation result based on the decryption key SK, to obtain cleartext data of the final computation result, and then store the cleartext data of the final computation result in the result storage system deployed locally.

[0127] According to another aspect, the target computing node may alternatively collect the ciphertext data transmitted by the other computing nodes and obtained by encrypting the respectively held private data, and perform computation on the collected ciphertext data transmitted by the other computing nodes and the ciphertext data generated locally together according to the appointed operation logic, to obtain ciphertext data corresponding to the final computation result obtained by performing a secure multi-party computation on the private data held by the computing nodes (the ciphertext data cannot be decrypted locally by the target computing node).

[0128] After the ciphertext data corresponding to the final computation result is obtained, the target computing node may broadcast the ciphertext data corresponding to the final computation result to the other computing nodes. After the other computing nodes receive the ciphertext data corresponding to the final computation result, the other computing nodes may decrypt the received ciphertext data corresponding to the final computation result based on the decryption key SK, to obtain cleartext data of the final computation result, and then store the cleartext data of the final computation result in the result storage system deployed locally.

[0129] It should be noted that, the operation logic appointed by the computing nodes still depends on an actual secure multi-party computation requirement, and this is not specifically limited in this specification. However, any operation logic may be decomposed into basic operations such as additive homomorphic encryption and multiplicative homomorphic encryption based on the homomorphic encryption algorithm.

[0130] 2) Garbled circuit algorithm

[0131] A principle of the garbled circuit algorithm is to simulate an application-specific

integrated circuit (ASIC) by using a computer to implement secure multi-party computation. Any secure multi-party computation task may be implemented by the ASIC designed for the task. That is, a gate circuit corresponding to the operation logic may be simulated by connecting a plurality of operation gates (basic operation gates such as AND gates and XOR gates, and the like) according to the operation logic corresponding to the secure multi-party computation task, and an operation is then performed by using the gate circuit.

[0132] Based on the garbled circuit algorithm, the operation logic corresponding to the secure multi-party computation task is generally represented as a gate circuit, each operation path of the gate circuit is then encrypted, and the encrypted gate circuit is referred to as a garbled circuit. Secure multi-party computation is implemented by eventually performing an operation of the garbled circuit by using a computer.

[0133] When secure multi-party computation is implemented based on the garbled circuit algorithm, computing nodes participating in the secure multi-party computation are generally classified into two roles, namely, a generation party of the garbled circuit and an execution party of the garbled circuit.

[0134] For example, in one secure multi-party computation in which K computing nodes participate, K-1 computing nodes are generally used as the generation party of a circuit to generate a garbled circuit, and one remaining computing node is used as the execution party of the garbled circuit to perform an operation on the garbled circuit, to obtain a final computation result.

[0135] If the secure multi-party computation algorithm carried in the computing nodes deployed in the distributed network is the garbled circuit algorithm, the operation logic corresponding to the secure multi-party computation task may first be represented as a gate circuit.

[0136] The operation logic requires that the plurality of computing nodes jointly participate based on the respectively held private data. One computing node may only correspond to one or more operation gates in the gate circuit, and the private data held by the computing node may only be used as inputted data of one or more operation gates in the gate circuit.

[0137] Further, after the target computing node reads the private data S needing to participate in the secure multi-party computation from a deployed data source, the target computing node may use the read private data S as an input value of a corresponding operation gate in the gate circuit to perform an operation in the operation gate, to obtain an output value of the operation gate.

[0138] In actual applications, the operation logic of each operation gate in the gate circuit

may be represented and performed by using

[0139] a truth table. One truth table is used for recording an output value obtained by an output path when any value is inputted into an input path of the operation gate. Each row in the truth table corresponds to one input value combination of the input path. For hiding true values of the operation paths in the operation gate in the operation of the gate circuit:

[0140] According to one aspect, based on existing operation logic of the garbled circuit algorithm, the target computing node may generally generate random numbers locally based on a random function carried in the garbled circuit algorithm, to replace the input values and the output values in the truth table of the operation gate, where the random numbers obtained after the replacement are referred to as salt values.

[0141] In this specification, the random numbers used for replacing the input values and the output values in the truth table of the operation gate may no longer be generated locally by the random function carried in the garbled circuit algorithm, but instead may be generated by a trusted key source deployed locally by the computing node instead.

[0142] In this case, the target computing node may obtain random numbers from the trusted key source, and replace the input values and the output values in the truth table based on the obtained random numbers.

[0143] According to another aspect, the target computing node may alternatively use the salt values corresponding to the input values of the operation gate as a key, to encrypt the salt values of the output values in the truth table of the operation gate to obtain ciphertext data, and replace the salt values of the output values in the original truth table based on the ciphertext data to obtain a garbled truth table; or the target computing node may independently create a garbled truth table based on the ciphertext data (that is, the garbled truth table only includes the ciphertext data).

[0144] According to a third aspect, the target computing node may alternatively disrupt a sequence of each row in the garbled truth table.

[0145] After the target computing node completes the garbled replacement for the truth table of the corresponding operation gate in the gate circuit in the foregoing three aspects, the garbled circuit is generated at the moment. The target computing node may construct an exchange message based on an existing message exchange mechanism of the garbled circuit. The target computing node uses the salt values corresponding to the input values of the corresponding operation gate in the gate circuit and the salt values corresponding to the input values of the operation gate as a key, and uses the ciphertext data obtained by encrypting the salt values of the output values in the truth table of the operation gate as a computing parameter, to transmit

the key and the computing parameter to the computing node used as the execution party of the garbled circuit.

[0146] For example, in actual applications, when secure multi-party computation is implemented based on the garbled circuit algorithm, the garbled circuit algorithm and oblivious transfer technology are generally combined. When the target computing node transmits the computing parameter to the computing node used as the execution party of the garbled circuit, on one hand, the target computing node may generally directly transmit the garbled truth table, in the form of a cleartext, to the computing node used as the execution party of the garbled circuit; on the other hand, the oblivious transfer technology may be used, to transmit, in a blurring transmission manner, the salt values corresponding to the input values of the operation gate corresponding to the target computing node in the gate circuit to the computing node used as the execution party of the garbled circuit. A specific implementation process of transmitting data to the computing node used as the execution party of the garbled circuit based on the oblivious transfer technology will not be repeatedly described in detail in this specification. When a person skilled in the art implements the technical solution of this specification, reference may be made to the record in the related art.

[0147] After receiving the computing parameters transmitted by the computing nodes, the computing node used as the execution party of the garbled circuit may perform decryption on the ciphertext data recorded in the garbled truth table of the operation gates corresponding to the computing nodes in the gate circuit according to the salt values corresponding to the input values of the operation gates corresponding to the computing nodes in the gate circuit, to obtain the salt values of the output values of the operation gates corresponding to the computing nodes in the gate circuit.

[0148] Eventually, the computing node used as the execution party of the garbled circuit may perform an operation grade by grade based on the eventually decrypted salt values of the output values of the operation gates in the gate circuit, to obtain a salt value corresponding to a final output value of the gate circuit, and then transmit the salt value as the final computation result to other computing nodes.

[0149] After receiving the salt value corresponding to the final output value of the gate circuit, the target computing node may query the truth table, to obtain an output value corresponding to the salt value based on a correspondence between a salt value and a true output value, and store the output value as an original value of a final computation result of the secure multi-party computation in the result storage system deployed locally.

[0150] For the computing nodes deployed in the distributed network, when the secure multi-

party computation process shown above is completed, the computing nodes may perform visual output on the computation result stored in the result storage system to administrative users of the computing nodes.

[0151] In this specification, an exchange message carrying a large number of ciphertexts is generally transmitted between computing nodes carrying secure multi-party computation algorithms such as the homomorphic encryption algorithm and the garbled circuit algorithm. Even for a user of the secure multi-party computation algorithm, it is generally difficult to understand the specific meanings of the ciphertexts carried in the exchange message.

[0152] Based on this, in this specification, when a target node device transmits, by constructing an exchange message based on the existing message exchange mechanism of the carried secure multi-party computation algorithm, the generated ciphertext data as a computing parameter to computing nodes participating in the secure multi-party computation, the target node device may further locally create an audit log related to the transmitted ciphertext data, and store the created audit log in a log storage system deployed locally.

[0153] The audit log may be, for example, used for recording description information related to the ciphertext data transmitted by the computing nodes participating in the secure multi-party computation.

[0154] In a shown implementation, the computing parameter that the computing node needs to transmit to other computing nodes participating in the secure multi-party computation is generally carried in a message field in the exchange message constructed by the computing node. Therefore, the specific meaning of each field and description information related to each field in the exchange message constructed by the computing node may be recorded in the audit log for auditing.

[0155] Correspondingly, when the target node device receives the computing parameters that are transmitted by the other computing nodes through the constructed exchange message and include at least the ciphertext data obtained by encrypting the held private data, the target node device may further locally create an audit log related to the ciphertext data transmitted by the other computing nodes, and store the created audit log in the log storage system deployed locally.

[0156] For example, the computing nodes deployed in the distributed network generally carry the same secure multi-party computation algorithm, so that the exchange message constructed by the computing nodes generally conforms to the same message format. Therefore, after receiving the exchange message transmitted by other computing nodes, the target computing node may parse the exchange message according to a standard message format to

generate a corresponding audit log, and record the specific meaning of each field in the exchange message in the audit log.

[0157] In this specification, the description information related to the ciphertext data and recorded in the audit log may cover any form of description information capable of proving that the encryption key corresponding to the ciphertext data is the trusted key from the trusted key source. This is not specifically limited in this specification.

[0158] In a shown implementation, the description information related to the ciphertext data and recorded in the audit log may include an encryption key corresponding to the ciphertext data. In this way, when a privacy audit is performed on the ciphertext data transmitted by the computing nodes based on the audit log stored in the log storage system, it may be directly audited whether the encryption key that is recorded in the audit log and corresponds to the ciphertext data is the trusted key from the trusted key source, to determine whether the ciphertext data contains the risk of privacy leakage.

[0159] In some scenarios, in addition to the encryption key, the audit log may further carry a decryption key used for decrypting the ciphertext data; and the audit log may even carry cleartext data of the ciphertext data.

[0160] For example, in an implementation, the description information related to the ciphertext data and recorded in the audit log created by the computing node and related to the ciphertext data transmitted locally to other computing nodes may be somewhat different from the description information related to the ciphertext data and recorded in the audit log created by the computing node and related to the received ciphertext data transmitted by the other computing nodes.

[0161] Due to the secure multi-party computing nodes, inputted data used by each computing node generally needs to be kept confidential from other computing nodes participating in the secure multi-party computation. Therefore, the audit log created by the computing node and related to the ciphertext data transmitted by the other computing nodes may only record the encryption key related to the ciphertext data. The audit log created by the computing node and related to the computing parameter transmitted locally to the other computing nodes may further record the decryption key of the ciphertext data and the cleartext data of the ciphertext data in addition to the encryption key of the ciphertext data.

[0162] In this way, an administrative user of the computing node may further verify, after the ciphertext data generated locally as a computing parameter by the computing node is decrypted based on the decryption key recorded in the audit log, whether decrypted cleartext data is consistent with the cleartext data of the computing parameter recorded in the audit log,

and audit whether the secure multi-party computation algorithm carried in the computing node maliciously adds private data to the transmitted ciphertext data without the user's knowledge, thereby improving the auditability of the algorithm.

[0163] In this specification, the computing nodes deployed in the distributed network may jointly use a secure multi-party algorithm provided by a provider of secure multi-party computation algorithms. Although the secure multi-party algorithm is an algorithm used for protecting private data, an incorrect or even malicious algorithm implementation is still prone to leakage of private data.

[0164] For example, in a scenario, a malicious secure multi-party computation algorithm may use a known key of other parties to encrypt private data needing to be transmitted to other computing nodes without a user's knowledge, to allow a receiver to obtain the private data held by the other computing nodes by using the known key to decrypt the data.

[0165] In this specification, the computing nodes deployed in the distributed network may perform a privacy audit on the secure multi-party algorithm carried in the computing nodes based on the audit log stored in the log storage system (or may only perform a privacy audit on the secure multi-party algorithm carried in this node or the secure multi-party algorithm carried in a specified computing node in the computing nodes), to determine whether the ciphertext data transmitted by the computing nodes contains the risk of privacy leakage.

[0166] Certainly, in actual applications, the privacy audit on the secure multi-party algorithm carried in the computing nodes may be alternatively performed by a third-party audit node. In this case, the computing nodes deployed in the distributed network may transmit the audit log in the log storage system to the audit node (for example, the audit log may be transmitted actively or the audit node may actively invoke the audit log), and the third-party audit node performs, based on the received audit log, a privacy audit on the secure multi-party algorithm carried in the computing nodes, to determine whether the computing parameters transmitted by the computing nodes contain the risk of privacy leakage.

[0167] For example, in a scenario, the computing nodes carry a secure multi-party computation algorithm provided by the provider, and audit nodes are independently deployed by operators of the computing nodes, where the audit nodes are used for performing a privacy audit on the secure multi-party computation algorithm provided by the provider.

[0168] The audit logic of performing a privacy audit on the secure multi-party algorithm carried in the computing nodes depends on specific content of the description information related to the ciphertext data and recorded in the audit log.

[0169] In a shown implementation, the description information related to the ciphertext data

and recorded in the audit log may include an encryption key corresponding to the ciphertext data.

[0170] In this scenario, the computing node may read an audit log corresponding to the ciphertext data transmitted as a computing parameter by the computing nodes participating in the secure multi-party computation from a log storage system deployed locally, and verify, based on the encryption key that is recorded in the read audit log and corresponds to the ciphertext data, whether the encryption key matches a trusted key from a deployed trusted key source.

[0171] For example, in actual applications, when a trusted key source generates a random number as a trusted key, a random number is generally randomly extracted as a trusted key from a preset random number value space in a random extraction manner. Therefore, in a case of verifying whether the encryption key that is recorded in the audit log and corresponds to the ciphertext data matches a trusted key from a deployed trusted key source, this process may be, for example, matching the value of a random number used as the encryption key against the random number value space, to determine whether the value of the random number of the encryption key is within a value range of the random number value space, if it does, it indicates that the value of the random number of the encryption key is a random number extracted from the value range of the random number value space, and the encryption key is a trusted key from a trusted key source in this case; otherwise, it indicates that the value of the random number of the encryption key is not a random number extracted from the value range of the random number value space, and the encryption key may not be a trusted key from a trusted key source in this case.

[0172] If it is determined through verification that the encryption key corresponding to the ciphertext data is a trusted key from a trusted key source, the encryption key corresponding to the ciphertext data is not a known key held by other computing nodes, and in this case, the ciphertext data does not contain the risk of privacy leakage; otherwise, if it is determined through verification that the encryption key corresponding to the ciphertext data is not a trusted key from a trusted key source, the encryption key corresponding to the ciphertext data may be a known key held by other computing nodes, and in this case, the ciphertext data contains the risk of privacy leakage.

[0173] In this way, the computing node may determine whether the key corresponding to the ciphertext data used as a computing parameter is a trusted key from a trusted key source by performing a simple audit on the audit log in the log storage system deployed locally, so that the risk of privacy leakage can be found in time.

[0174] Certainly, in some scenarios, if the audit log further carries a decryption key used for decrypting the ciphertext data and cleartext data of the ciphertext data in addition to the encryption key, in this case, the computing node may further perform a security verification on the ciphertext data based on the decryption key corresponding to the ciphertext data and the cleartext data of the ciphertext data that are recorded in the read audit log, to determine whether private data is maliciously added to the ciphertext data by a carried secure multi-party computation algorithm.

[0175] For example, in actual applications, after the ciphertext data generated locally as a computing parameter by the computing node is decrypted based on the decryption key recorded in the audit log, the computing node may further verify whether the decrypted cleartext data of the computing parameter is consistent with the cleartext data of the computing parameter recorded in the audit log, and audit whether the secure multi-party computation algorithm carried in the computing node maliciously adds private data to the ciphertext data transmitted as a computing parameter without the user's knowledge. If they are consistent, it indicates that the implementation logic of the secure multi-party computation algorithm that is carried in the computing node and is provided by a provider is the same as the design of the secure multi-party computation algorithm, and the ciphertext data does not contain the risk of privacy leakage. If they are not consistent, it indicates that the secure multi-party computation algorithm that is carried in the computing node and is provided by an algorithm provider may maliciously add private data to the transmitted computing parameter without the user's knowledge, and in this case, the ciphertext data contains the risk of privacy leakage.

[0176] In this way, the computing node may determine, by performing a simple audit on the audit log in the log storage system deployed locally, whether private data is maliciously added to the ciphertext data used as a computing parameter by a carried secure multi-party algorithm, so that the risk of privacy leakage can be found in time.

[0177] According to the foregoing technical solution, the computing nodes participating in the secure multi-party computation may obtain a trusted key from the connected trusted key source, encrypt the held private data based on the obtained trusted key to obtain ciphertext data, and then transmit at least the ciphertext data as a computing parameter to other computing nodes participating in the secure multi-party computation to perform the secure multi-party computation. In this way, while performing encrypted computation on the held private data, the computing node may no longer use a key generated locally by using a carried secure multi-party computation algorithm, and use the trusted key obtained from the connected trusted key source instead, so that the risk of privacy leakage caused by constructing a key according to a

malicious secure multi-party computation algorithm based on the held private data to encrypt the private data can be avoided.

[0178] Corresponding to the foregoing method embodiment, this specification further provides an embodiment of a secure multi-party computation apparatus. The embodiment of the secure multi-party computation apparatus in this specification may be applicable to an electronic device. The apparatus embodiment may be implemented by software, or by hardware or a combination of software and hardware. Using a software implementation as an example, as a logical apparatus, the apparatus is formed by reading corresponding computer program instructions from a non-transitory storage into a memory by a processor of an electronic device where the apparatus is located. On a hardware level, as shown in FIG. 2, which is a hardware structural diagram of an electronic device in which a secure multi-party computation apparatus according to this specification is located, the electronic device in which the apparatus is located in the embodiment not only includes a processor, a memory, a network interface, and a non-transitory storage shown in FIG. 2, but usually may also include other hardware according to actual functions of the electronic device. Details will not be repeated herein.

[0179] FIG. 3 is a block diagram of a secure multi-party computation apparatus according to an exemplary embodiment of this specification.

[0180] Referring to FIG. 3, the secure multi-party computation apparatus 30 may be applicable to the electronic device shown in FIG. 2, and includes an obtaining module 301, an encryption module 302, and a transmission module 303.

[0181] The obtaining module 301 is configured to obtain a trusted key from the trusted key source.

[0182] The encryption module 302 is configured to encrypt the held private data based on the obtained trusted key to obtain ciphertext data.

[0183] The transmission module 303 is configured to transmit at least the ciphertext data as a computing parameter to other computing nodes participating in the secure multi-party computation for the other computing nodes to perform the secure multi-party computation based on collected computing parameters transmitted by the computing nodes participating in the secure multi-party computation.

[0184] In this embodiment, the apparatus 30 further includes:

[0185] a creation module 304 (not shown in FIG. 3), configured to create an audit log corresponding to the ciphertext data transmitted by the computing node to the other computing nodes, the audit log recording description information related to the ciphertext data.

[0186] In this embodiment, the apparatus 30 further includes:

[0187] a collection module 305 (not shown in FIG. 3), configured to collect computing parameters transmitted by the other computing nodes participating in the secure multi-party computation, the computing parameters including at least ciphertext data obtained by encrypting the private data held by the other computing nodes; and

[0188] a computation module 306 (not shown in FIG. 3), configured to perform the secure multi-party computation based on the collected computing parameters, where

[0189] the creation module 304 is further configured to:

[0190] create an audit log corresponding to the ciphertext data transmitted by the other computing nodes, the audit log including description information related to the encrypted private data.

[0191] In this embodiment, the apparatus 30 further includes:

[0192] an audit module 307, configured to perform, based on the audit log, a privacy audit on the ciphertext data transmitted by the computing node participating in the secure multi-party computation, to determine whether the ciphertext data transmitted as a computing parameter by the computing node contains a risk of privacy leakage; or

[0193] transmit the audit log to a third-party audit node for the audit node to perform a privacy audit on the ciphertext data transmitted as a computing parameter by the computing node participating in the secure multi-party computation, to determine whether the ciphertext data transmitted as a computing parameter by the computing node contains a risk of privacy leakage.

[0194] In this embodiment, the description information related to the ciphertext data includes an encryption key corresponding to the ciphertext data; and

[0195] the audit module 307 is further configured to:

[0196] verify whether the encryption key that is recorded in the audit log and corresponds to the ciphertext data transmitted by the computing node participating in the secure multi-party computation matches the trusted key from the trusted key source; and

[0197] if it does, determine that the ciphertext data transmitted as the computing parameter by the computing node does not contain the risk of privacy leakage; or

[0198] if it does not, determine that the ciphertext data transmitted as the computing parameter by the computing node contains the risk of privacy leakage.

[0199] In this embodiment, a decryption key corresponding to the ciphertext data and cleartext data corresponding to the ciphertext data are further recorded in the audit log corresponding to the ciphertext data transmitted by the computing node to the other computing nodes; and

[0200] the audit module 307 is further configured to:

[0201] decrypt, based on the decryption key that is recorded in the audit log and corresponds to the ciphertext data transmitted by the computing node participating in the secure multi-party computation, the ciphertext data to obtain cleartext data;

[0202] verify whether the decrypted cleartext data matches the cleartext data that is recorded in the audit log and corresponds to the ciphertext data; and

[0203] if it does, determine that the ciphertext data transmitted as the computing parameter by the computing node does not contain the risk of privacy leakage; or

[0204] if it does not, determine that the ciphertext data transmitted as the computing parameter by the computing node contains the risk of privacy leakage.

[0205] In this embodiment, the trusted key is a random number; and

[0206] the obtaining the trusted key from the trusted key source includes any one of the following:

[0207] obtaining a random number generated by the trusted key source as the trusted key;

[0208] obtaining a random number generated by the trusted key source, and using a result obtained by further performing an operation on the random number generated by the trusted key source and a random number generated by the computing node as the trusted key;

[0209] obtaining a random number generated by the trusted key source, and using a random number calculated by inputting the random number into a key generation function as the trusted key; and

[0210] obtaining a random number generated by the trusted key source, and using a result obtained by further performing an operation on a random number calculated by inputting the random number into a key generation function and a random number generated by the computing node as the trusted key.

[0211] For details about the implementation processes of the functions and effects of the modules in the foregoing apparatus, refer to the implementation processes of the corresponding steps in the foregoing method. Details are not described herein again.

[0212] The apparatus embodiment basically corresponds to the method embodiment, and therefore for related parts, reference may be made to partial descriptions in the method embodiment. The 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 plurality of network modules. A part or all of the modules may be selected according to actual requirements to achieve the objectives of the solutions of this specification.

A person of ordinary skill in the art may understand and implement the solutions without creative efforts.

[0213] The system, apparatus, module, or unit described in the foregoing embodiment may be, for example, implemented by a computer chip or entity, or a product with a specified function. A typical implementation device is a computer, and a specific form of the computer may be a personal computer, a laptop computer, a cellular phone, a camera phone, a smartphone, a personal digital assistant, a media player, a navigation device, an email device, a game console, a tablet computer, a wearable device, or a combination of any of these devices.

[0214] Corresponding to the foregoing method embodiment, this specification further provides an embodiment of an electronic device. The electronic device includes a processor and a memory configured to store machine-executable instructions, where the processor and the memory are generally connected by an internal bus. In other possible implementations, the device may further include an external interface for communication with other devices or components.

[0215] In this embodiment, by reading and executing the machine-executable instructions stored in the memory and corresponding to control logic of secure multi-party computation, the processor is configured to:

[0216] obtain a trusted key from the trusted key source;

[0217] encrypt the held private data based on the obtained trusted key to obtain ciphertext data; and

[0218] transmit at least the ciphertext data as a computing parameter to other computing nodes participating in the secure multi-party computation for the other computing nodes to perform the secure multi-party computation based on collected computing parameters transmitted by the computing nodes participating in the secure multi-party computation.

[0219] In this embodiment, by reading and executing the machine-executable instructions stored in the memory and corresponding to control logic of secure multi-party computation, the processor is configured to:

[0220] create an audit log corresponding to the ciphertext data transmitted by the computing node to the other computing nodes, the audit log recording description information related to the ciphertext data.

[0221] In this embodiment, by reading and executing the machine-executable instructions stored in the memory and corresponding to control logic of secure multi-party computation, the processor is configured to:

[0222] collect computing parameters transmitted by the other computing nodes participating

in the secure multi-party computation, the computing parameters including at least ciphertext data obtained by encrypting the private data held by the other computing nodes;

[0223] perform the secure multi-party computation based on the collected computing parameters; and

[0224] create an audit log corresponding to the ciphertext data transmitted by the other computing nodes, the audit log including description information related to the encrypted private data.

[0225] In this embodiment, by reading and executing the machine-executable instructions stored in the memory and corresponding to control logic of secure multi-party computation, the processor is configured to:

[0226] perform, based on the audit log, a privacy audit on the ciphertext data transmitted by the computing node participating in the secure multi-party computation, to determine whether the ciphertext data transmitted as a computing parameter by the computing node contains the risk of privacy leakage.

[0227] In this embodiment, the description information related to the ciphertext data includes an encryption key corresponding to the ciphertext data; and

[0228] by reading and executing the machine-executable instructions stored in the memory and corresponding to control logic of secure multi-party computation, the processor is configured to:

[0229] verify whether the encryption key that is recorded in the audit log and corresponds to the ciphertext data transmitted by the computing node participating in the secure multi-party computation matches the trusted key from the trusted key source; and

[0230] if it does, determine that the ciphertext data transmitted as the computing parameter by the computing node does not contain the risk of privacy leakage; or

[0231] if it does not, determine that the ciphertext data transmitted as the computing parameter by the computing node contains the risk of privacy leakage.

[0232] In this embodiment, a decryption key corresponding to the ciphertext data and cleartext data corresponding to the ciphertext data are further recorded in the audit log corresponding to the ciphertext data transmitted by the computing node to the other computing nodes; and

[0233] by reading and executing the machine-executable instructions stored in the memory and corresponding to control logic of secure multi-party computation, the processor is configured to:

[0234] decrypt, based on the decryption key that is recorded in the audit log and corresponds

to the ciphertext data transmitted by the computing node participating in the secure multi-party computation, the ciphertext data to obtain cleartext data;

[0235] verify whether the decrypted cleartext data matches the cleartext data that is recorded in the audit log and corresponds to the ciphertext data; and

[0236] if it does, determine that the ciphertext data transmitted as the computing parameter by the computing node does not contain the risk of privacy leakage; or

[0237] if it does not, determine that the ciphertext data transmitted as the computing parameter by the computing node contains the risk of privacy leakage.

[0238] Other embodiments of this specification will be apparent to a person skilled in the art from consideration of the specification and practice of the present application disclosed here. The present disclosure is intended to cover any variations, uses, or adaptations of the present disclosure following the general principles thereof and including such departures from the present disclosure as come within known or customary practice in the art. It is intended that the specification and examples be considered as exemplary only, with a true scope and spirit of the present disclosure being indicated by the following claims.

[0239] It will be appreciated that the present disclosure is not limited to the exact construction that has been described above and illustrated in the accompanying drawings, and that various modifications and changes can be made without departing from the scope thereof. It is intended that the scope of this specification only be limited by the appended claims.

[0240] The foregoing descriptions are merely exemplary embodiments of this specification, but are not intended to limit this specification. Any modification, equivalent replacement, or improvement made without departing from the spirit and principle of this specification should fall within the protection scope of this specification.

CLAIMS

What is claimed is:

1. A secure multi-party computation method, applicable to one computing node of a plurality of computing nodes deployed in a distributed network, the plurality of computing nodes jointly participating in a secure multi-party computation based on data held on each of the plurality of computing nodes,, the one computing node coupled to a trusted key source, the method comprising:

obtaining, by the one computing node, a trusted key from the trusted key source;

encrypting, by the one computing node and based on the trusted key, the data held by the one computing node to obtain internal ciphertext data; and

transmitting, by the one computing node, an internal computing parameter comprising at least the internal ciphertext data to other computing nodes of the plurality of computing nodes for the other computing nodes to perform the secure multi-party computation based on the internal computing parameter;

collecting, by the one computing node, an external computing parameter transmitted by one of the other computing nodes participating in the secure multi-party computation, the external computing parameter comprising at least external ciphertext data obtained by encrypting data held by the one of the other computing nodes;

creating a first audit log corresponding to the external ciphertext data, the first audit log comprising description information related to the external ciphertext data, wherein the description information comprises an encryption key corresponding to the external ciphertext data;

performing, based on the first audit log, a privacy audit on the external ciphertext data to determine whether the external ciphertext data contains a privacy risk by:

determining whether the encryption key in the first audit log matches the trusted key from the trusted key source, by:

determining, without decrypting the external ciphertext data, whether the encryption key in the first audit log is within a value range of a preset random number value space,

determining, in response to a determination that the encryption key is within the value range, that the encryption key matches the trusted key, and

determining, in response to a determination that the encryption key is not within the value range, that the encryption key does not match the trusted key; and

determining, in response to the encryption key matching the trusted key from the trusted source, that the external ciphertext data does not contain a privacy risk; or

determining, in response to the encryption key not matching the trusted key from the trusted source, that the external ciphertext data contains a privacy risk.

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

creating, by the one computing node, a second audit log corresponding to the internal ciphertext data, the second audit log comprising description information related to the internal ciphertext data.

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

performing, by the one computing node, the secure multi-party computation based on the external computing parameter.

4. The method according to claim 3, further comprising:

transmitting the first audit log corresponding to the external ciphertext data to a third-party audit node for the third-party audit node to perform a privacy audit on the external ciphertext data to determine whether the external ciphertext data contains a privacy risk.

5. The method according to claim 2, wherein the second audit log corresponding to the internal ciphertext data further comprises a decryption key corresponding to the internal ciphertext data and original cleartext data corresponding to the internal ciphertext data, and the method further comprises:

decrypting, based on the decryption key that is recorded in the second audit log corresponding to the internal ciphertext data, the internal ciphertext data to obtain decrypted cleartext data; and

verifying whether the decrypted cleartext data matches the original cleartext data; and

determining, in response to the decrypted cleartext data matching the original cleartext data, that the internal ciphertext data does not contain a privacy risk; or

determining, in response to the decrypted cleartext data not matching the original cleartext data, that the internal ciphertext data contains a privacy risk.

6. The method according to claim 1, wherein the trusted key is a number; and

the obtaining a trusted key from the trusted key source comprises any one of the following:

obtaining an initial random number generated by the trusted key source, and using the initial random number as the trusted key;

obtaining an initial random number generated by the trusted key source, obtaining a computation random number generated by the one computing node, and using a result obtained by further performing an operation on the initial random number computation random number as the trusted key;

obtaining an initial random number generated by the trusted key source, and obtaining a functional number by inputting the initial random number into a key generation function, and using the functional number as the trusted key; and

obtaining an initial random number generated by the trusted key source, obtaining a functional number by inputting the initial random number into a key generation function, obtaining a computation random number generated by the one computing node, and using a result obtained by further performing an operation on the computation random number and the functional number as the trusted key.

7. The method of claim 1, wherein the distributed network is a blockchain network, and the plurality of computing nodes are node devices in the blockchain network.

8. A secure multi-party computation apparatus, comprising a processor and a non-transitory computer-readable memory coupled to the processor, and configured with instructions executable by the processor to perform operations, comprising:

obtaining, by one computing node of a plurality of computing nodes, a trusted key from a trusted key source coupled to the one computing node, wherein the plurality of computing nodes deployed in a distributed network and jointly participating in a secure multi-party computation based on data held on each of the plurality of computing nodes;

encrypting, by the one computing node and based on the trusted key, the data held by the one computing node to obtain internal ciphertext data;

transmitting, by the one computing node, an internal computing parameter comprising at least the internal ciphertext data to other computing nodes of the plurality of computing nodes for the other computing nodes to perform the secure multi-party computation based on the internal computing parameter;

collecting, by the one computing node, an external computing parameter transmitted by one of the other computing nodes participating in the secure multi-party computation, the external computing parameter comprising at least external ciphertext data obtained by

encrypting data held by the one of the other computing nodes;

creating a first audit log corresponding to the external ciphertext data, the first audit log comprising description information related to the external ciphertext data, wherein the description information comprises an encryption key corresponding to the external ciphertext data;

performing, based on the first audit log, a privacy audit on the external ciphertext data to determine whether the external ciphertext data contains a privacy risk by:

determining whether the encryption key in the first audit log matches the trusted key from the trusted key source, by:

determining, without decrypting the external ciphertext data, whether the encryption key in the first audit log is within a value range of a preset random number value space,

determining, in response to a determination that the encryption key is within the value range, that the encryption key matches the trusted key, and

determining, in response to a determination that the encryption key is not within the value range, that the encryption key does not match the trusted key; and

determining, in response to the encryption key matching the trusted key from the trusted source, that the external ciphertext data does not contain a privacy risk; or

determining, in response to the encryption key not matching the trusted key from the trusted source, that the external ciphertext data contains a privacy risk.

9. The apparatus of claim 8, wherein the operations further comprise:

creating, by the one computing node, a second audit log corresponding to the internal ciphertext data, the second audit log comprising description information related to the internal ciphertext data.

10. The apparatus of claim 9, wherein the operations further comprise:

performing, by the one computing node, the secure multi-party computation based on the external computing parameter.

11. The apparatus of claim 10, wherein the operations further comprise:

transmitting the first audit log corresponding to the external ciphertext data to a third-party audit node for the third-party audit node to perform a privacy audit on the external ciphertext data to determine whether the external ciphertext data contains a privacy risk.

12. The apparatus of claim 9, wherein the second audit log corresponding to the internal ciphertext data further comprises a decryption key corresponding to the internal ciphertext data and original cleartext data corresponding to the internal ciphertext data, and wherein the operations further comprise:

decrypting, based on the decryption key in the second audit log corresponding to the internal ciphertext data, the internal ciphertext data to obtain decrypted cleartext data; and

verifying whether the decrypted cleartext data matches the original cleartext data; and

determining, in response to the decrypted cleartext data matching the original cleartext data, that the internal ciphertext data does not contain a privacy risk; or

determining, in response to the decrypted cleartext data not matching the original cleartext data, that the internal ciphertext data contains a privacy risk.

13. The apparatus of claim 8, wherein the trusted key is a number, and wherein the obtaining a trusted key from the trusted key source comprises any one of the following:

obtaining an initial random number generated by the trusted key source, and using the initial random number as the trusted key;

obtaining an initial random number generated by the trusted key source, obtaining a computation random number generated by the one computing node, and using a result obtained by further performing an operation on the initial random number and the computation random number as the trusted key;

obtaining an initial random number generated by the trusted key source, and obtaining a functional number by inputting the initial random number into a key generation function, and using the functional number as the trusted key; and

obtaining an initial random number generated by the trusted key source, obtaining a functional number by inputting the initial random number into a key generation function, obtaining a computation random number generated by the one computing node, using a result obtained by further performing an operation on the computation random number and the functional number as the trusted key.

14. The apparatus of claim 8, wherein the distributed network is a blockchain network, and the plurality of computing nodes are node devices in the blockchain network.

15. A non-transitory computer-readable storage medium having instructions stored thereon executable by a processor to cause the processor to perform operations comprising:

- obtaining, by one computing node of a plurality of computing nodes, a trusted key from a trusted key source coupled to the one computing node, wherein the plurality of computing nodes deployed in a distributed network and jointly participating in a secure multi-party computation based on data held on each of the plurality of computing nodes;

- encrypting, by the one computing node and based on the trusted key, the data held by the one computing node to obtain internal ciphertext data;

- transmitting, by the one computing node, an internal computing parameter comprising at least the internal ciphertext data to other computing nodes of the plurality of computing nodes for the other computing nodes to perform the secure multi-party computation based on the internal computing parameter;

- collecting, by the one computing node, an external computing parameter transmitted by one of the other computing nodes participating in the secure multi-party computation, the external computing parameter comprising at least external ciphertext data obtained by encrypting data held by the one of the other computing nodes;

- creating a first audit log corresponding to the external ciphertext data, the first audit log comprising description information related to the external ciphertext data, wherein the description information comprises an encryption key corresponding to the external ciphertext data;

- performing, based on the first audit log, a privacy audit on the external ciphertext data to determine whether the external ciphertext data contains a privacy risk by:

 - determining whether the encryption key in the first audit log matches the trusted key from the trusted key source, by:

 - determining, without decrypting the external ciphertext data, whether the encryption key in the first audit log is within a value range of a preset random number value space,

 - determining, in response to a determination that the encryption key is within the value range, that the encryption key matches the trusted key, and

 - determining, in response to a determination that the encryption key is not within the value range, that the encryption key does not match the trusted key; and

 - determining, in response to the encryption key matching the trusted key from the trusted source, that the external ciphertext data does not contain a privacy risk; or

 - determining, in response to the encryption key not matching the trusted key from the

trusted source, that the external ciphertext data contains a privacy risk.

16. The storage medium of claim 15, wherein the operations further comprise:

creating, by the one computing node, a second audit log corresponding to the internal ciphertext data, the second audit log comprising description information related to the internal ciphertext data.

17. The storage medium of claim 16, wherein the operations further comprise:

performing, by the one computing node, the secure multi-party computation based on the external computing parameter.

18. The storage medium of claim 17, wherein the operations further comprise:

transmitting the first audit log corresponding to the external ciphertext data to a third-party audit node for the third-party audit node to perform a privacy audit on the external ciphertext data to determine whether the external ciphertext data contains a privacy risk.