

# **SECURE MULTI-PARTY COMPUTATION METHOD AND APPARATUS, AND ELECTRONIC DEVICE**

## **TECHNICAL FIELD**

**[0001]** One or more embodiments of this specification relate 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 one computing node deployed in a distributed network, a plurality of computing nodes being deployed in the distributed network, and 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 random source, the method including:

**[0005]** obtaining a trusted random number from the trusted random source;

**[0006]** performing an operation on the held private data based on the obtained trusted random number to obtain an operation result; and

**[0007]** transmitting at least the trusted random number 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 trusted random number transmitted by the computing node to the other computing nodes, the audit log recording description information related to the trusted random number.

**[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 at least including random numbers transmitted 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 random numbers transmitted by the other computing nodes, the audit log including description information related to the random numbers.

**[0014]** Optionally, the method further includes:

**[0015]** performing, based on the audit log, a privacy audit on the random number transmitted as a computing parameter by the computing node participating in the secure multi-party computation, to determine whether the random number transmitted as a computing parameter by the computing node is at 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 random number transmitted as a computing parameter by the computing node participating in the secure multi-party computation, to determine whether the random number transmitted as a computing parameter by the computing node is at a risk of privacy leakage.

**[0017]** Optionally, the description information related to the trusted random number includes a random number generation rule corresponding to the trusted random number; and

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

**[0019]** verifying, based on the random number generation rule that is recorded in the audit log and corresponds to the random number transmitted by the computing node participating in

the secure multi-party computation, whether the random number matches the trusted random number from the trusted random source; and

**[0020]** if it does, determining that the random number transmitted as the computing parameter by the computing node is at the risk of privacy leakage; or

**[0021]** if it does not, determining that the random number transmitted as the computing parameter by the computing node is not at the risk of privacy leakage.

**[0022]** Optionally, the obtaining a trusted random number from the trusted random source includes any one of the following:

**[0023]** obtaining a random number generated by the trusted random source as the trusted random number;

**[0024]** obtaining a random number generated by the trusted random source, and using a result obtained by further performing an operation on the random number generated by the trusted random source and a random number generated by the computing node as the trusted random number;

**[0025]** obtaining a random number generated by the trusted random source, and using a random number computed by inputting the random number generated by the trusted random source into a key generation function as the trusted random number; and

**[0026]** obtaining a random number generated by the trusted random source, and using a result obtained by further performing an operation on a random number computed by inputting the random number generated by the trusted random source into a key generation function and a random number generated by the computing node as the trusted random number.

**[0027]** This specification further provides a secure multi-party computation apparatus, applicable to any one computing node deployed in a distributed network, a plurality of computing nodes being deployed in the distributed network, and 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 random source, the apparatus including:

**[0028]** an obtaining module, configured to obtain a trusted random number from the trusted random source;

**[0029]** an operation module, configured to perform an operation on the held private data based on the obtained trusted random number to obtain an operation result; and

**[0030]** a transmission module, configured to transmit at least the trusted random number 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.

**[0031]** Optionally, the method further includes:

**[0032]** a creation module, configured to create an audit log corresponding to the trusted random number transmitted by the computing node to the other computing nodes, the audit log recording description information related to the trusted random number.

**[0033]** Optionally, the method further includes:

**[0034]** a collection module, configured to collect computing parameters transmitted by the other computing nodes participating in the secure multi-party computation, the computing parameters at least including random numbers transmitted by the other computing nodes;

**[0035]** perform the secure multi-party computation based on the collected computing parameters; and

**[0036]** create an audit log corresponding to the random numbers transmitted by the other computing nodes, the audit log including description information related to the random numbers.

**[0037]** Optionally, the apparatus further includes:

**[0038]** an audit module, configured to perform, based on the audit log, a privacy audit on the random number transmitted as a computing parameter by the computing node participating in the secure multi-party computation, to determine whether the random number transmitted as a computing parameter by the computing node is at a risk of privacy leakage; or

**[0039]** transmit the audit log to a third-party audit node for the audit node to perform a privacy audit on the random number transmitted as a computing parameter by the computing node participating in the secure multi-party computation, to determine whether the random number transmitted as a computing parameter by the computing node is at a risk of privacy leakage.

**[0040]** Optionally, the description information related to the trusted random number includes a random number generation rule corresponding to the trusted random number; and

**[0041]** the audit module is further configured to:

**[0042]** verify, based on the random number generation rule that is recorded in the audit log and corresponds to the random number transmitted by the computing node participating in the secure multi-party computation, whether the random number matches the trusted random number from the trusted random source; and

**[0043]** if it does, determine that the random number transmitted as the computing parameter by the computing node is at the risk of privacy leakage; or

[0044] if it does not, determine that the random number transmitted as the computing parameter by the computing node is not at the risk of privacy leakage.

[0045] Optionally, the obtain a trusted random number from the trusted random source includes any one of the following:

[0046] obtain a random number generated by the trusted random source as the trusted random number;

[0047] obtain a random number generated by the trusted random source, and use a result obtained by further performing an operation on the random number generated by the trusted random source and a random number generated by the computing node as the trusted random number;

[0048] obtain a random number generated by the trusted random source, and use a random number computed by inputting the random number generated by the trusted random source into a key generation function as the trusted random number; and

[0049] obtain a random number generated by the trusted random source, and use a result obtained by further performing an operation on a random number computed by inputting the random number generated by the trusted random source into a key generation function and a random number generated by the computing node as the trusted random number.

[0050] This specification further provides an electronic device, including:

[0051] a processor; and

[0052] a memory configured to store machine-executable instructions, where

[0053] 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:

[0054] obtain a trusted random number from the trusted random source;

[0055] perform an operation on held private data based on the obtained trusted random number to obtain an operation result; and

[0056] transmit at least the trusted random number 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.

[0057] According to the foregoing technical solution, the computing nodes participating in the secure multi-party computation may obtain a trusted random number from the connected trusted random source, perform an operation on the held private data based on the obtained trusted random number to obtain an operation result, and then transmit at least the trusted

random number 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, the computing node may no longer use a random number generated locally by using a carried secure multi-party computation algorithm as a computing parameter, and use the trusted random number obtained from the connected trusted random source as the computing parameter instead, so that the risk of privacy leakage caused by constructing a pseudo-random number according to a malicious secure multi-party computation algorithm based on the held private data as a computing parameter can be avoided.

## **BRIEF DESCRIPTION OF THE DRAWINGS**

**[0058]** FIG. 1 is a flowchart of a secure multi-party computation method according to an exemplary embodiment.

**[0059]** FIG. 2 is a schematic structural diagram of an electronic device according to an exemplary embodiment.

**[0060]** FIG. 3 is a block diagram of a secure multi-party computation apparatus according to an exemplary embodiment.

## **DETAILED DESCRIPTION OF THE INVENTION**

**[0061]** For a computing node carrying secure multi-party computation algorithms such as a secret sharing algorithm and a garbled circuit algorithm, one or more random numbers are generally generated based on carried randomized algorithms, the generated random number is used to perform an operation on held private data to obtain an operation result, and the generated random number and the foregoing operation result are then transmitted as computing parameters to other computing nodes participating in secure multi-party computation.

**[0062]** However, in actual applications, in a malicious secure multi-party algorithm, the private data held by the computing node participating in the secure multi-party computation may be used to construct a pseudo-random number, and the pseudo-random number and an operation result obtained by performing an operation on the held private data are then transmitted based on the pseudo-random number to the other computing nodes participating in the secure multi-party computation. In this way, the other computing nodes may perform a backward recovery on the pseudo-random number, to restore the original private data used to construct the pseudo-random number, leading to privacy leakage.

**[0063]** For example, the held private data may be inputted into a reversible random number

generation function to obtain a pseudo-random number; and a receiver may perform inverse computation based on the pseudo-random number by using an inverse function of the random number generation function to restore the original private data used to construct the pseudo-random number.

**[0064]** Based on this, this specification is intended to provide a technical solution of using a trusted random number from a trusted random source as a computing parameter of secure multi-party computation, to avoid privacy leakage caused by constructing a pseudo-random number as a computing parameter based on private data held by a computing node participating in the secure multi-party computation.

**[0065]** During implementation, computing nodes in a distributed network may further be connected to a trusted random source respectively in addition to carrying a secure multi-party computation algorithm provided by a provider, where the foregoing trusted random sources may be random sources independently developed by operators of the computing nodes or may be random sources developed by a trusted third party other than the foregoing provider.

**[0066]** When the computing node participates in secure multi-party computation based on held private data, the computing node may obtain a trusted random number from the connected trusted random source, and perform an operation on the held private data based on the trusted random number to obtain an operation result; and then transmit at least the foregoing trusted random number as a computing parameter (in some scenarios, the foregoing operation result may also be used as a computing parameter together) to other computing nodes participating in the secure multi-party computation in the distributed network, so that the other computing nodes perform secure multi-party computation based on collected computing parameters transmitted by the computing nodes participating in the secure multi-party computation.

**[0067]** According to the foregoing technical solution, the computing nodes participating in the secure multi-party computation may obtain a trusted random number from the connected trusted random source, perform an operation on the held private data based on the obtained trusted random number to obtain an operation result, and then transmit at least the trusted random number 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, the computing node may no longer use a random number generated locally by using a carried secure multi-party computation algorithm as a computing parameter, and use the trusted random number obtained from the connected trusted random source as the computing parameter instead, so that the risk of privacy leakage caused by constructing a pseudo-random number according to a malicious secure multi-party computation algorithm based on the held

private data as a computing parameter can be avoided.

**[0068]** This specification is described below with reference to specific embodiments and specific application scenarios.

**[0069]** FIG. 1 shows a secure multi-party computation method according to an embodiment of this specification, applicable to any one computing node deployed in a distributed network, a plurality of computing nodes being deployed in the distributed network, and 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 random source, and the secure multi-party computation method performs the following steps:

**[0070]** Step 102: Obtain a trusted random number from the trusted random source.

**[0071]** Step 104: Perform an operation on the held private data based on the obtained trusted random number to obtain an operation result.

**[0072]** Step 106: Transmit at least the trusted random number 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.

**[0073]** The distributed network may include a data network in any form that can support secure multi-party computation and uses a distributed architecture.

**[0074]** For example, in one scenario, the distributed network may be a secure multi-party computation platform constructed by a device cluster (such as a server cluster) formed by a plurality of computing devices.

**[0075]** 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.

**[0076]** The computing node may specifically include a node device that can carry the secure multi-party computation algorithm in the distributed network.

**[0077]** The secure multi-party computation algorithm may specifically 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 specifically correspond to secure multi-party computation software independently developed by the computing nodes or centrally provided by a provider.

**[0078]** The computing parameters are specifically 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 computing parameters generated through an exchange message and mutual transmission between the computing nodes to perform secure multi-party computation based on collected computing parameters generated by the computing nodes.

**[0079]** In actual applications, the computing parameter may generally include random numbers needing to participate in secure multi-party computation; and/or encrypted data needing to participate in secure multi-party computation.

**[0080]** 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 or a garbled circuit algorithm. 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.

**[0081]** 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 N<sup>th</sup> random number) obtained by performing an operation on the N-1 random numbers and the held private data.

**[0082]** Based on the homomorphic encryption algorithm, the computing parameter generally includes encrypted data obtained by encrypting the private data based on a held key.

**[0083]** 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 encrypted data obtained by encrypting the salt value of outputted data by using the salt value of the input value as a key.

**[0084]** The trusted random source may include random sources independently developed by operators of the computing nodes, or may include random sources provided by a trusted third party other than the foregoing provider. For example, in actual applications, the trusted random source may be specifically a computer program (for example, a trusted random number

generation algorithm or a trusted random number generation function) independently developed by the operators of the computing nodes and used for generating a trusted random source, or may be a computer program developed by a trusted third party other than the provider and used for generating a trusted random source.

**[0085]** The operation result includes an operation result obtained by performing an irreversible operation on the trusted random number and the held private data; and the held private data cannot be restored by performing an inverse operation on the operation result.

**[0086]** Using the secret sharing algorithm as an example, the operation result is the  $N^{\text{th}}$  random number obtained by performing an operation on the  $N-1$  random numbers and the held private data together.

**[0087]** Using the garbled circuit algorithm as another example, the operation result is the encrypted data obtained by encrypting the salt value of the outputted data by using the salt value corresponding to the input value of the garbled circuit as a key after the input value of the garbled circuit is replaced based on the trusted random number.

**[0088]** 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 secure multi-party computation without leaking the respectively held private data.

**[0089]** 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 encrypted data participating in the secure multi-party computation between the computing nodes, to participate in secure multi-party computation.

**[0090]** It should be noted that, in this specification, the secure multi-party computation algorithm carried in the computing nodes is specifically an algorithm that supports transmission of the encrypted 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.

**[0091]** In addition to jointly carrying the same type of secure multi-party computation algorithm, a trusted random 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.

**[0092]** The trusted random source is used for generating a trusted random number.

**[0093]** The data source is used for storing and maintaining private data held by data nodes.

**[0094]** The result storage system is used for storing and maintaining a computation result eventually obtained by participating in secure multi-party computation by the data nodes.

**[0095]** It should be noted that, in actual applications, alternatively, the trusted random source may be specifically a random source provided by a trusted third party other than the provider of the secure multi-party computation algorithm. In this case, alternatively, a trusted random source may not be deployed locally in the computing nodes, but instead a third-party trusted random source is directly connected to the computing nodes, to obtain a trusted random source used for performing an operation on the held private data.

**[0096]** 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 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.

**[0097]** The log storage system is used for storing and maintaining audit logs created by the data nodes and related to the computing parameters.

**[0098]** 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.

**[0099]** 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 random number from the trusted random source.

**[0100]** The target computing node may then perform an operation on the read private data based on the obtained trusted random number, to generate an operation result corresponding to the held private data.

**[0101]** In an implementation shown, 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 the computed random number as the trusted key.

**[0102]** In another implementation shown, 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.

**[0103]** An operation rule used for performing an operation on the read private data based on the obtained trusted random number generally depends on a specific type of a secure multi-party computation algorithm carried by the node device. For different secure multi-party computation algorithms, operation rules used for performing an operation on the read private data based on the obtained trusted random number may be different.

**[0104]** For example, if the secure multi-party computation algorithm carried by the node device is the secret sharing algorithm, the operation rule may specifically correspond to the process of obtaining the  $N^{\text{th}}$  random number by performing an operation on the  $N-1$  random numbers and the held private data together.

**[0105]** In another example, if the secure multi-party computation algorithm carried by the node device is the garbled circuit algorithm, the operation rule may specifically correspond to the process of encrypting the salt value of the outputted data by using the salt value corresponding to the input value of the garbled circuit as a key after the input value of the garbled circuit is replaced based on the trusted random number.

**[0106]** After the target computing node performs an operation on the read private data based on the obtained trusted random number to generate an operation result 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 trusted random number as a computing parameter to other computing nodes participating in secure multi-party computation.

**[0107]** It should be noted that, in actual applications, the trusted random number and the operation result may be transmitted together as computing parameters to the other computing

nodes participating in secure multi-party computation.

**[0108]** Using an example in which the secure multi-party computation algorithm carried in the computing node is the secret sharing algorithm, based on the secret sharing algorithm, the operation result is an  $N^{\text{th}}$  share obtained by splitting the private data. In this case, the computing node may choose to reserve the  $N^{\text{th}}$  share locally, and transmit the remaining  $N-1$  random numbers as the computing parameters to other computing nodes, or may choose to reserve one of the  $N-1$  random numbers locally, and transmit the  $N^{\text{th}}$  random number and the remaining random numbers as the computing parameters to other computing nodes.

**[0109]** The other computing nodes may collect computing parameters transmitted, through the exchange message, 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.

**[0110]** Correspondingly, the target computing node may alternatively collect computing parameters transmitted, through the exchange message, 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.

**[0111]** 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.

**[0112]** 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 secret sharing algorithm and the garbled circuit algorithm.

**[0113]** 1) Secret sharing algorithm

**[0114]** The secret sharing algorithm is also referred to as a secret splitting algorithm. A basic principle of the secret sharing algorithm is to split a secret (for example, a key) into a plurality of shares and deliver the shares to different people for storage. The secret can be restored only when a quantity of collected shares exceeds a threshold quantity. If the quantity of the collected shares is less than the threshold quantity, any information about the secret cannot be restored based on the shares. When the secret sharing algorithm is applied to secure multi-party computation, the threshold quantity is generally the same as a quantity of computing nodes

needing to participate in the secure multi-party computation and deployed in a distributed network.

**[0115]** When the secret sharing algorithm is applied to the secure multi-party computation, the computing nodes respectively split inputted data into shares and exchange the shares with each other. The computing nodes then respectively perform an operation on locally collected shares to obtain a share (that is, one share of the final computation result) corresponding to a final computation result of the secure multi-party computation. Eventually, the computing nodes merge all generated shares corresponding to the final computation result to obtain the final computation result, to complete the secure multi-party computation.

**[0116]** If the secure multi-party computation algorithm carried in the computing nodes deployed in the distributed network is the secret sharing algorithm, after the target computing node reads private data  $S$  needing to participate in the secure multi-party computation from a deployed data source, the target computing node may first split the private data into  $d$  shares (random numbers) according to the secret sharing algorithm.

**[0117]** Generally, based on existing computation logic of the secret sharing algorithm and the secret sharing algorithm,  $d-1$  independent random numbers are generated in a preset random number value space  $P$  in a random extraction manner as the first  $d-1$  shares, and the shares are recorded as  $S_1, S_2, S_3, \dots, S_{d-1}$ .

**[0118]** For example, if the private data  $S$  is integer data of 64 bits,  $P = 2^{64}$ ; and the  $d-1$  independent random numbers generated eventually are all values extracted from a value space of  $[0, 2^{64}-1]$ .

**[0119]** In this specification, the first  $d-1$  shares may no longer be generated according to the secret sharing algorithm, but instead may be generated by trusted random sources deployed locally in the computing nodes.

**[0120]** Further, the target computing node may obtain the first  $d-1$  trusted random numbers from the trusted random sources, perform an operation on the first  $d-1$  random numbers and the private data  $S$  together, and use an operation result obtained through the operation as a  $d^{\text{th}}$  share, which is recorded as  $S_d$ .

**[0121]** An operation rule for performing an operation on the  $d-1$  random numbers and the private data  $S$  together generally depends on a specific secret sharing algorithm.

**[0122]** For example, the most common trivial additive secret sharing algorithm in the secret sharing algorithm is used as an example, and  $d$  shares obtained by splitting the private data are generally required to meet the following expression:

**[0123]**  $S = S_1 + S_2 + S_3 + \dots + S_d \% P$ ,

**[0124]** where % is the modulus operator.

**[0125]** Based on the foregoing expression,  $S_d = S - (S_1 + S_2 + S_3 + \dots + S_{d-1}) \% P$ . In this algorithm,  $S_d$  may be obtained by subtracting a modulus computation result of a sum of the first  $d-1$  shares and the value space  $P$  from the private data  $S$ .

**[0126]** After data splitting for the private data  $S$  is completed based on the secret sharing algorithm, the target computing node may reserve one share locally, construct an exchange message based on an existing message exchange mechanism, and transmit the remaining shares to other computing nodes participating in the secure multi-party computation respectively.

**[0127]** The target computing node may reserve the  $N^{\text{th}}$  share locally, or may reserve any one of the first  $d-1$  shares. That is, when the target computing node reserves the  $N^{\text{th}}$  share locally, only the random numbers used as the first  $d-1$  shares are eventually used as the computing parameters. When the target computing node reserves any one of the first  $d-1$  shares locally, the remaining random numbers used as the first  $d-1$  shares and an operation result of the  $N-1$  random numbers and the private data  $S$  are used as the computing parameters.

**[0128]** According to one aspect, other computing nodes may collect shares transmitted by the computing nodes, perform computation on the collected shares according to operation logic appointed by the computing nodes to obtain a share corresponding to the final computation result, and transmit the computation result to the other computing nodes for broadcasting. The target computing node may collect the computation results broadcast by the other computing nodes, perform computation on the collected computation results again according to the appointed operation logic to obtain a final computation result obtained by performing secure multi-party computation on the private data held by the computing nodes, and then store the final computation result in the result storage system deployed locally.

**[0129]** According to another aspect, the target computing node may alternatively collect the shares transmitted by the other computing nodes, perform computation on the collected shares according to the appointed operation logic to obtain a share corresponding to the final computation result, and transmit the computation result to the other computing nodes for broadcasting. The other computing nodes may collect the computation results broadcast by the computing nodes, and perform computation on the collected computation results again according to the appointed operation logic to obtain a final result obtained by performing secure multi-party computation on the held private data  $S$ .

**[0130]** The operation logic appointed by the computing nodes generally depends on an

actual secure multi-party computation requirement, and this is not specifically limited in this specification.

**[0131]** 2) Garbled circuit algorithm

**[0132]** 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) according to the operation logic corresponding to the secure multi-party computation task, and an operation is then performed by using the gate circuit.

**[0133]** 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.

**[0134]** 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.

**[0135]** 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.

**[0136]** 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.

**[0137]** The operation logic requires that the plurality of computing nodes to jointly participate based on the respectively held private data. One computing node may only correspond to one or several 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 several operation gates in the gate circuit.

**[0138]** Further, after the target computing node reads the private data S needing to

participate in 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.

**[0139]** In actual applications, the operation logic of each operation gate in the gate circuit may be represented and performed by using

**[0140]** 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:

**[0141]** 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.

**[0142]** 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 random source deployed locally by the computing node.

**[0143]** In this case, the target computing node may obtain random numbers from the trusted random source, and replace the input values and the output values in the truth table based on the obtained random numbers.

**[0144]** 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 perform asymmetric encryption on the salt values of the output values in the truth table of the operation gate to obtain encrypted data, and replace the salt values of the output values in the original truth table based on the encrypted data to obtain a garbled truth table; or the target computing node may independently create a garbled truth table based on the encrypted data (that is, the garbled truth table only includes the encrypted data).

**[0145]** According to a third aspect, the target computing node may alternatively disrupt a sequence of each row in the garbled truth table.

**[0146]** 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 encrypted data obtained by performing asymmetric encryption on 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 serving as the execution party of the garbled circuit.

**[0147]** 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 serving 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 clear text, to the computing node serving 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 serving as the execution party of the garbled circuit.

**[0148]** A specific implementation process of transmitting data to the computing node serving 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.

**[0149]** After receiving the computing parameters transmitted by the computing nodes, the computing node serving as the execution party of the garbled circuit may perform decryption on the encrypted 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.

**[0150]** Eventually, the computing node serving 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.

**[0151]** 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.

**[0152]** 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.

**[0153]** In this specification, an exchange message carrying a large quantity of random numbers is generally transmitted between computing nodes carrying secure multi-party computation algorithms such as the secret sharing 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 random numbers carried in the exchange message.

**[0154]** 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 carried secure multi-party computation algorithm, the obtained trusted random number and the foregoing operation result as computing parameters 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 random number, and store the created audit log in a log storage system deployed locally.

**[0155]** The audit log may be specifically used for recording description information related to the random numbers transmitted by the computing nodes participating in the secure multi-party computation.

**[0156]** In an implementation shown, 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 carried 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.

**[0157]** Correspondingly, when the target node device receives at least the random numbers transmitted as computing parameters by other computing nodes through the constructed exchanged message, the target node device may further create an audit log related to the random numbers transmitted by the other computing nodes, and store the created audit log in the log

storage system deployed locally.

**[0158]** 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.

**[0159]** In this specification, the description information related to the random number and recorded in the audit log may cover any form of description information capable of proving that the random number is from the trusted random number of the trusted random source. This is not specifically limited in this specification.

**[0160]** In an implementation shown, the description information related to the random number and recorded in the audit log may specifically include a random number generation rule corresponding to the random number.

**[0161]** The random number generation rule may specifically cover any form of description information related to a generation mode of the random number.

**[0162]** For example, the random number is a trusted random number obtained from a trusted random source, a random number obtained by further performing an operation on a trusted random number obtained from a trusted random source and a random number generated locally, a random number generated by inputting a random number seed obtained from a trusted random source into a random number generation function, a random number generated by inputting a random number seed obtained from a trusted random source into a random number generation function, a random number obtained by further performing an operation on a random number generated locally or the like, which is not listed one by one in this specification.

**[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 construct a pseudo-random number based on private data held by the computing node without a user's knowledge, and transmit the private data to other computing nodes by using the pseudo-random number.

**[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 description information related to the random number and recorded in 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 random numbers transmitted by the computing nodes are at 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 a privacy audit on the secure multi-party algorithm carried in the computing nodes based on the received audit log, to determine whether the computing parameters transmitted by the computing nodes are at 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 random number and recorded in the audit log.

**[0169]** In an implementation shown, the description information related to the encrypted data and recorded in the audit log may specifically include a random number generation rule corresponding to the encrypted data.

**[0170]** In this scenario, the computing node may read an audit log corresponding to a random number transmitted as a computing parameter by the computing node participating in secure multi-party computation from a log storage system deployed locally, and check, based on a random number generation rule that is recorded in the read audit log and corresponds to the random number, whether the random number is a trusted random number from the trusted random source.

**[0171]** A specific check rule generally fully corresponds to the random number generation rule.

**[0172]** For example, in an implementation, if the random number generation rule is a trusted

random number obtained from a trusted random source, because when a trusted key source generates a random number as a trusted key, a random number is generally extracted as a trusted key from a preset random number value space in a random extraction manner, the check rule may be specifically: determining whether the value of the random number is within a value range of the random number value space, if it does, it indicates that the random number is a random number extracted from the value range of the random number value space, and the random number is a trusted key from a trusted key source in this case; otherwise, it indicates that the random number is not a random number extracted from the value range of the random number value space, and the random number may not be a trusted key from a trusted key source in this case.

**[0173]** Correspondingly, in another implementation, if the random number generation rule is a random number generated by inputting a random number seed obtained from a trusted random source into a random number generation function, the check rule may be specifically checking whether the random number seed is from a valid data source of a trusted random source.

**[0174]** If it is determined through the check that the random number is a trusted random number from a trusted random source or a data source generating the random number is a valid data source from trusted random sources, the random number used as a computing parameter is not at the risk of privacy leakage; or otherwise, if it is determined through the check that the random number is not a trusted random number from a trusted random source, or the data source generating the random number is not a valid data source from trusted random sources, the random number may be a pseudo-random number constructed based on the private data held by the computing node in this case, and the random number as a computing parameter is at the risk of privacy leakage.

**[0175]** In this way, the computing node can determine whether the random number used as a computing parameter is a trusted random number from a trusted random 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.

**[0176]** According to the foregoing technical solution, the computing nodes participating in the secure multi-party computation may obtain a trusted random number from the connected trusted random source, perform an operation on the held private data based on the obtained trusted random number to obtain an operation result, and then transmit at least the trusted random number and the operation result as computing parameters to other computing nodes participating in the secure multi-party computation to perform the secure multi-party

computation. In this way, the computing node may no longer use a random number generated locally by using a carried secure multi-party computation algorithm as a computing parameter, and use the trusted random number obtained from the connected trusted random source as the computing parameter instead, so that the risk of privacy leakage caused by constructing a pseudo-random number according to a malicious secure multi-party computation algorithm based on the held private data as a computing parameter can be avoided.

**[0177]** 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 applied 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 are not described herein again.

**[0178]** FIG. 3 is a block diagram of a secure multi-party computation apparatus according to an exemplary embodiment of this specification.

**[0179]** Referring to FIG. 3, the secure multi-party computation apparatus 30 may be applied to the electronic device shown in FIG. 1, and includes an obtaining module 301, an operation module 302, and a transmission module 303.

**[0180]** The obtaining module 301 is configured to obtain a trusted random number from a trusted random source.

**[0181]** The operation module 302 is configured to perform an operation on held private data based on the obtained trusted random number to obtain an operation result.

**[0182]** The transmission module 303 is configured to transmit at least the trusted random number 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.

**[0183]** In this embodiment, the apparatus 30 further includes:

[0184] a creation module 304 (not shown in FIG. 3), configured to create an audit log corresponding to the trusted random number transmitted by the computing node to the other computing nodes, the audit log recording description information related to the trusted random number.

[0185] In this embodiment, the apparatus 30 further includes:

[0186] 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 at least including random numbers transmitted by the other computing nodes;

[0187] perform the secure multi-party computation based on the collected computing parameters; and

[0188] create an audit log corresponding to the random numbers transmitted by the other computing nodes, the audit log including description information related to the random numbers.

[0189] In this embodiment, the apparatus 30 further includes:

[0190] an audit module 306 (not shown in FIG. 3), configured to perform, based on the audit log, a privacy audit on the random number transmitted as a computing parameter by the computing node participating in the secure multi-party computation, to determine whether the random number transmitted as a computing parameter by the computing node is at a risk of privacy leakage; or

[0191] transmit the audit log to a third-party audit node for the audit node to perform a privacy audit on the random number transmitted as a computing parameter by the computing node participating in the secure multi-party computation, to determine whether the random number transmitted as a computing parameter by the computing node is at a risk of privacy leakage.

[0192] In this embodiment, the description information related to the trusted random number includes a random number generation rule corresponding to the trusted random number; and

[0193] the audit module 306 is further configured to:

[0194] verify, based on the random number generation rule that is recorded in the audit log and corresponds to the random number transmitted by the computing node participating in the secure multi-party computation, whether the random number matches the trusted random number from the trusted random source; and

[0195] if it does, determine that the random number transmitted as a computing parameter by the computing node is at the risk of privacy leakage; or

**[0196]** if it does not, determine that the random number transmitted as a computing parameter by the computing node is not at the risk of privacy leakage.

**[0197]** In this embodiment, the obtain a trusted random number from the trusted random source includes any one of the following:

**[0198]** obtain a random number generated by the trusted random source as the trusted random number;

**[0199]** obtain a random number generated by the trusted random source, and use a result obtained by further performing an operation on the random number generated by the trusted random source and a random number generated by the computing node as the trusted random number;

**[0200]** obtain a random number generated by the trusted random source, and use a random number computed by inputting the random number generated by the trusted random source into a key generation function as the trusted random number; and

**[0201]** obtain a random number generated by the trusted random source, and use a result obtained by further performing an operation on a random number computed by inputting the random number generated by the trusted random source into a key generation function and a random number generated by the computing node as the trusted random number.

**[0202]** 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 will not be repeated herein.

**[0203]** 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.

**[0204]** The system, apparatus, module, or unit described in the foregoing embodiment may be specifically 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.

**[0205]** 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.

**[0206]** 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:

**[0207]** obtain a trusted random number from the trusted random source;

**[0208]** perform an operation on held private data based on the obtained trusted random number to obtain an operation result; and

**[0209]** transmit at least the trusted random number 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.

**[0210]** 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:

**[0211]** create an audit log corresponding to the trusted random number transmitted by the computing node to the other computing nodes, the audit log recording description information related to the trusted random number.

**[0212]** 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:

**[0213]** collect computing parameters transmitted by the other computing nodes participating in the secure multi-party computation, the computing parameters at least including random numbers transmitted by the other computing nodes;

**[0214]** perform the secure multi-party computation based on the collected computing parameters; and

**[0215]** create an audit log corresponding to the random numbers transmitted by the other computing nodes and store the audit log, the audit log including description information related to the random numbers.

**[0216]** 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:

**[0217]** perform, based on the audit log, a privacy audit on the random number transmitted as a computing parameter by the computing node participating in the secure multi-party computation, to determine whether the random number transmitted as a computing parameter by the computing node is at a risk of privacy leakage; or

**[0218]** transmit the audit log to a third-party audit node for the audit node to perform a privacy audit on the random number transmitted as a computing parameter by the computing node participating in the secure multi-party computation, to determine whether the random number transmitted as a computing parameter by the computing node is at a risk of privacy leakage.

**[0219]** In this embodiment, the description information related to the trusted random number includes a random number generation rule corresponding to the trusted random number; and

**[0220]** 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:

**[0221]** verify, based on the random number generation rule that is recorded in the audit log and corresponds to the random number transmitted by the computing node participating in the secure multi-party computation, whether the random number matches the trusted random number from the trusted random source; and

**[0222]** if it does, determine that the random number transmitted as a computing parameter by the computing node is at the risk of privacy leakage; or

**[0223]** if it does not, determine that the random number transmitted as a computing parameter by the computing node is not at the risk of privacy leakage.

**[0224]** Other embodiments of this specification will be apparent to those skilled in the art from consideration of the specification and practice of the present application disclosed here. This specification is intended to cover any variations, uses, or adaptations of this specification following the general principles thereof and including such departures from this specification 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 this specification being indicated by the following claims.

**[0225]** It will be appreciated that this specification 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.

**[0226]** 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 any one computing node deployed in a distributed network, a plurality of computing nodes being deployed in the distributed network, and 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 random source, the method comprising:

obtaining a trusted random number from the trusted random source;

performing an operation on the held private data based on the obtained trusted random number to obtain an operation result; and

transmitting at least the trusted random number 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.

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

creating an audit log corresponding to the trusted random number transmitted by the computing node to the other computing nodes, the audit log recording description information related to the trusted random number.

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

collecting computing parameters transmitted by the other computing nodes participating in the secure multi-party computation, the computing parameters at least comprising random numbers transmitted by the other computing nodes;

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

creating an audit log corresponding to the random numbers transmitted by the other computing nodes, the audit log comprising description information related to the random numbers.

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

performing, based on the audit log, a privacy audit on the random number transmitted as a computing parameter by the computing node participating in the secure multi-party

computation, to determine whether the random number transmitted as a computing parameter by the computing node is at a risk of privacy leakage; or

transmitting the audit log to a third-party audit node for the audit node to perform a privacy audit on the random number transmitted as a computing parameter by the computing node participating in the secure multi-party computation, to determine whether the random number transmitted as a computing parameter by the computing node is at a risk of privacy leakage.

5. The method according to claim 4, wherein the description information related to the trusted random number comprises a random number generation rule corresponding to the trusted random number; and

the performing, based on the audit log, a privacy audit on the random number transmitted as a computing parameter by the computing node participating in the secure multi-party computation, to determine whether the random number transmitted as a computing parameter by the computing node is at a risk of privacy leakage comprises:

verifying, based on the random number generation rule that is recorded in the audit log and corresponds to the random number transmitted by the computing node participating in the secure multi-party computation, whether the random number matches the trusted random number from the trusted random source; and

if it does, determining that the random number transmitted as the computing parameter by the computing node is at the risk of privacy leakage; or

if it does not, determining that the random number transmitted as the computing parameter by the computing node is not at the risk of privacy leakage.

6. The method according to claim 1, wherein the obtaining a trusted random number from the trusted random source comprises any one of the following:

obtaining a random number generated by the trusted random source as the trusted random number;

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

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

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

7. A secure multi-party computation apparatus, applicable to any one computing node deployed in a distributed network, a plurality of computing nodes being deployed in the distributed network, and 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 random source, the apparatus comprising:

- an obtaining module, configured to obtain a trusted random number from the trusted random source;

- an operation module, configured to perform an operation on the held private data based on the obtained trusted random number to obtain an operation result; and

- a transmission module, configured to transmit at least the trusted random number 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.

8. The apparatus according to claim 7, further comprising:

- a creation module, configured to create an audit log corresponding to the trusted random number transmitted by the computing node to the other computing nodes, the audit log recording description information related to the trusted random number.

9. The method according to claim 8, further comprising:

- a collection module, configured to collect computing parameters transmitted by the other computing nodes participating in the secure multi-party computation, the computing parameters at least comprising random numbers transmitted by the other computing nodes;

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

- create an audit log corresponding to the random numbers transmitted by the other computing nodes, the audit log comprising description information related to the random numbers.

10. The apparatus according to claim 8 or 9, further comprising:

an audit module, configured to perform, based on the audit log, a privacy audit on the random number transmitted as a computing parameter by the computing node participating in the secure multi-party computation, to determine whether the random number transmitted as a computing parameter by the computing node is at a risk of privacy leakage; or

transmit the audit log to a third-party audit node for the audit node to perform a privacy audit on the random number transmitted as a computing parameter by the computing node participating in the secure multi-party computation, to determine whether the random number transmitted as a computing parameter by the computing node is at a risk of privacy leakage.

11. The apparatus according to claim 10, wherein the description information related to the trusted random number comprises a random number generation rule corresponding to the trusted random number; and

the audit module is further configured to:

verify, based on the random number generation rule that is recorded in the audit log and corresponds to the random number transmitted by the computing node participating in the secure multi-party computation, whether the random number matches the trusted random number from the trusted random source; and

if it does, determine that the random number transmitted as the computing parameter by the computing node is at the risk of privacy leakage; or

if it does not, determine that the random number transmitted as the computing parameter by the computing node is not at the risk of privacy leakage.

12. The apparatus according to claim 7, wherein the obtaining a trusted random number from the trusted random source comprises any one of the following:

obtaining a random number generated by the trusted random source as the trusted random number;

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

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

obtaining a random number generated by the trusted random source, and using a result

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

13. An electronic device, comprising:
  - a processor; and
  - a memory configured to store machine-executable instructions, wherein 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:
    - obtain a trusted random number from the trusted random source;
    - perform an operation on held private data based on the obtained trusted random number to obtain an operation result; and
    - transmit at least the trusted random number 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.