US 20240146505A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2024/0146505 A1**

Tsuchida (43) **Pub. Date: May 2, 2024**

(54) **SECURE COMPUTATION SYSTEM, SECURE COMPUTATION SERVER APPARATUS, SECURE COMPUTATION METHOD, AND SECURE COMPUTATION PROGRAM**

(71) Applicant: **NEC Corporation**, Minato-ku, Tokyo (JP)

(72) Inventor: **Hikaru Tsuchida**, Tokyo (JP)

(73) Assignee: **NEC Corporation**, Minato-ku, Tokyo (JP)

(21) Appl. No.: **18/272,733**

(22) PCT Filed: **Jan. 18, 2021**

(86) PCT No.: **PCT/JP2021/001468**

§ 371 (c)(1),
(2) Date: **Jul. 17, 2023**

Publication Classification

(51) **Int. Cl.**
H04L 9/06 (2006.01)

(52) **U.S. Cl.**
CPC ***H04L 9/0631*** (2013.01); *H04L 2209/08* (2013.01)

(57) **ABSTRACT**

An secure computation server apparatus in a secure computation system includes: a local shuffle part that computes, by using a shared permutation shared by four of the five secure computation server apparatuses, permuted values of a share for a remaining one of the five secure computation server apparatuses and sends the permuted values of the share to the remaining secure computation server apparatus; a comparison and verification part that compares values with each other, which are received from at least three of the four secure computation server apparatuses and which are supposed to be a same value, and adopts the values that are same at least two values as an accurate permutation; and a shuffle synthesis part that synthesizes mini-shuffles, by using a shared permutation shared by a corresponding combination of four secure computation server apparatuses and a permutation adopted by a corresponding one of the comparison and verification parts.

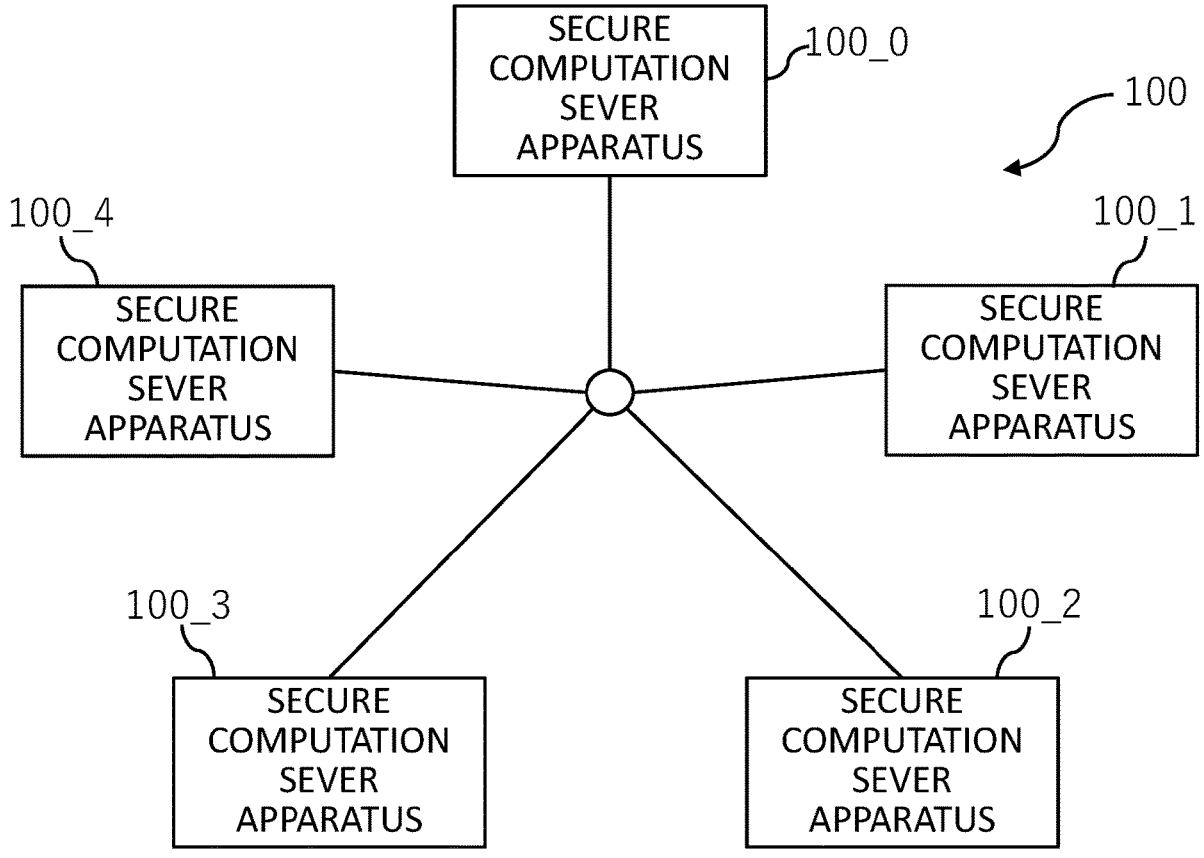


FIG. 1

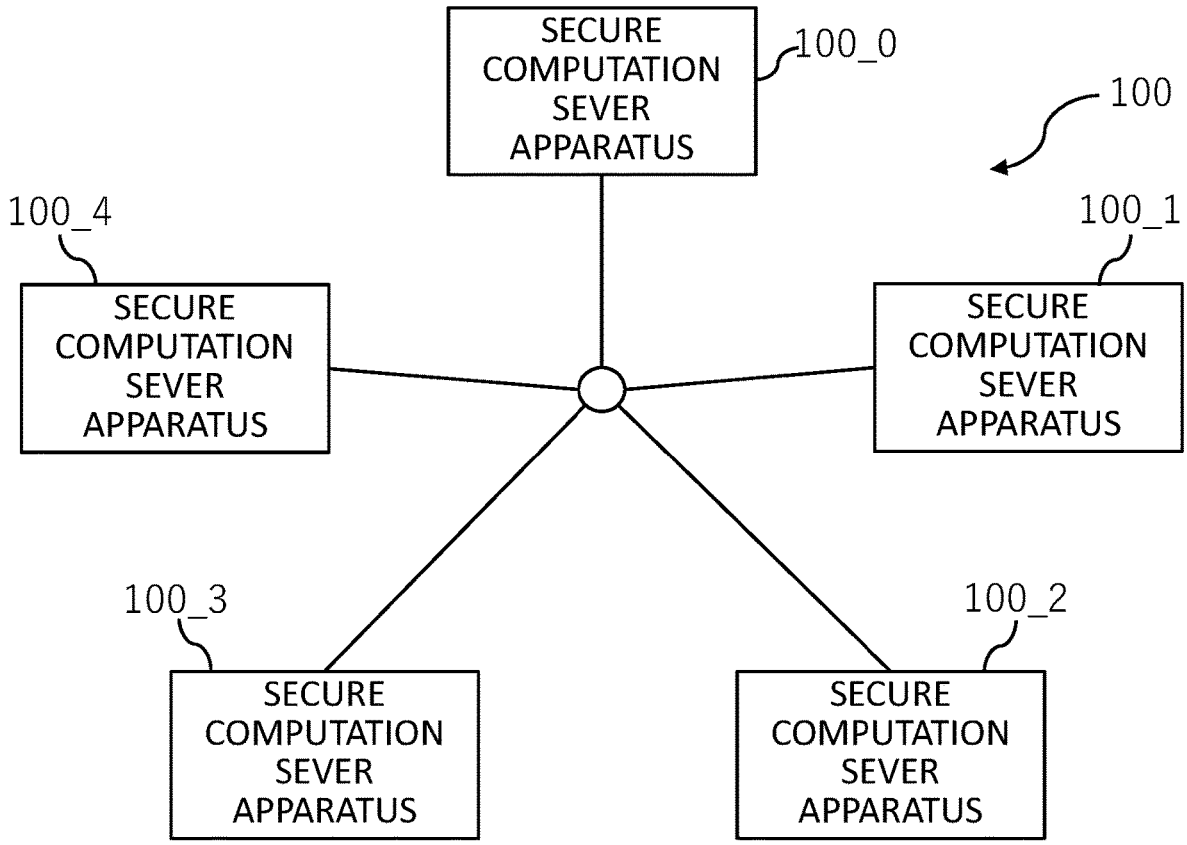

SECURE COMPUTATION SEVER APPARATUS
100_0
100
100_4
SECURE COMPUTATION SEVER APPARATUS
100_1
SECURE COMPUTATION SEVER APPARATUS
100_3
SECURE COMPUTATION SEVER APPARATUS
100_2
SECURE COMPUTATION SEVER APPARATUS

FIG. 2

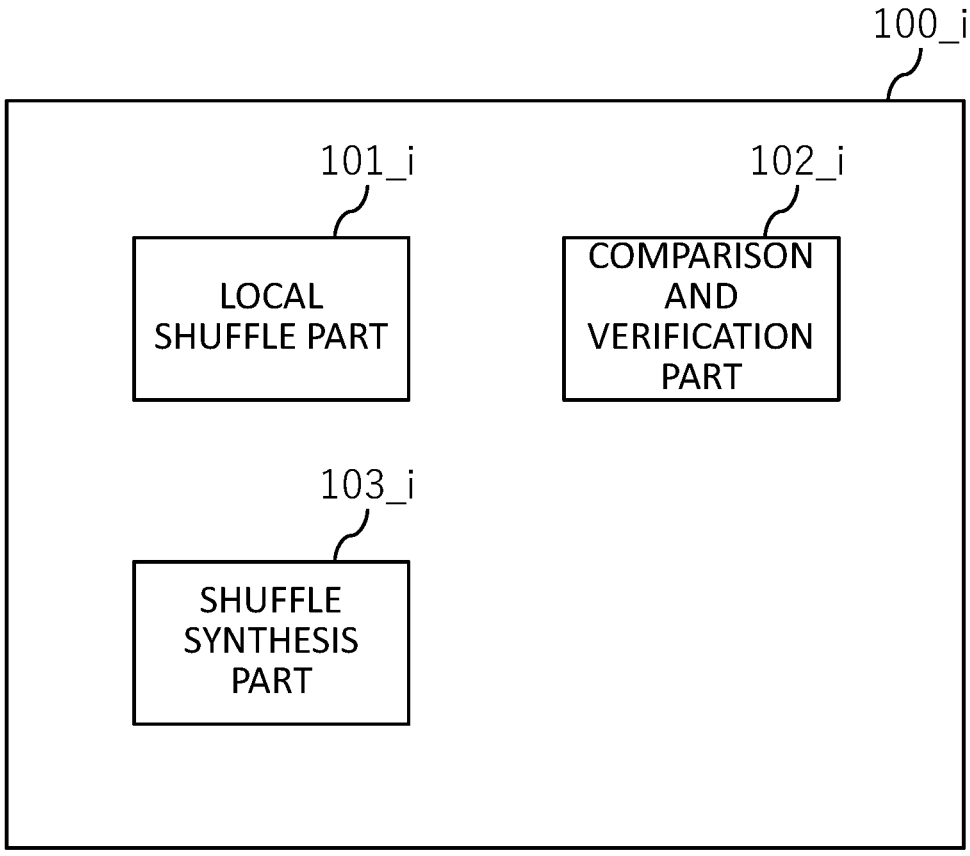

100_i
101_i
LOCAL SHUFFLE PART
102_i
COMPARISON AND VERIFICATION PART
103_i
SHUFFLE SYNTHESIS PART

FIG. 3

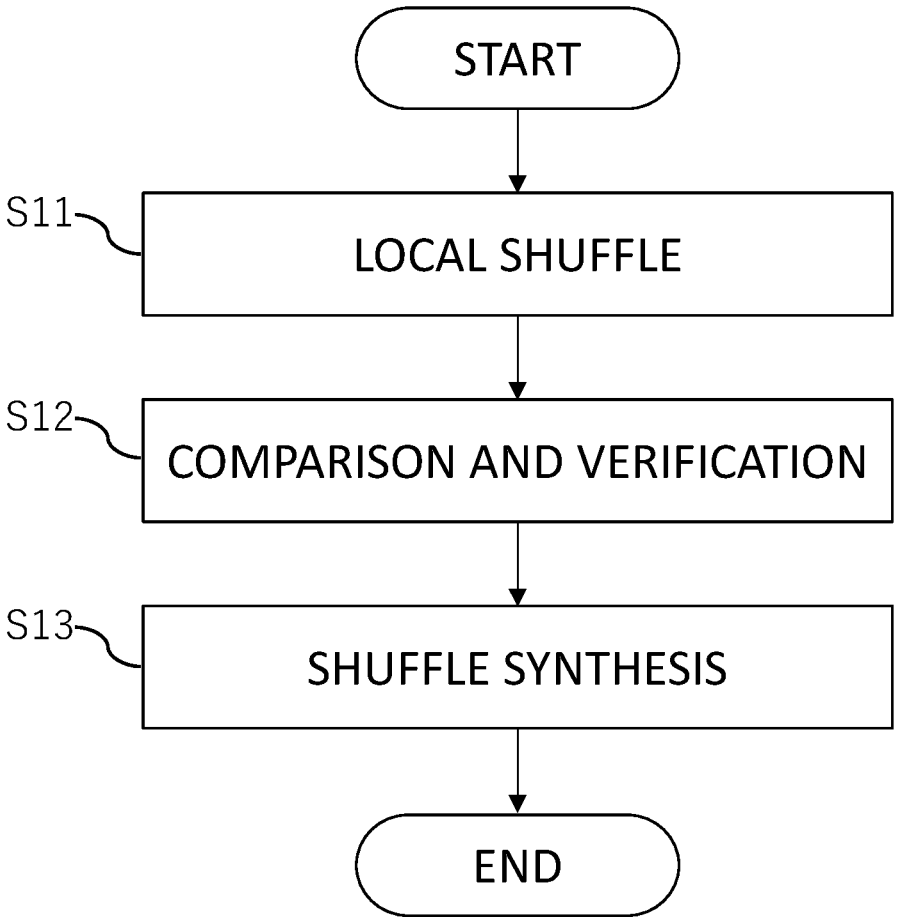

START
S11
LOCAL SHUFFLE
S12
COMPARISON AND VERIFICATION
S13
SHUFFLE SYNTHESIS
END

FIG. 4

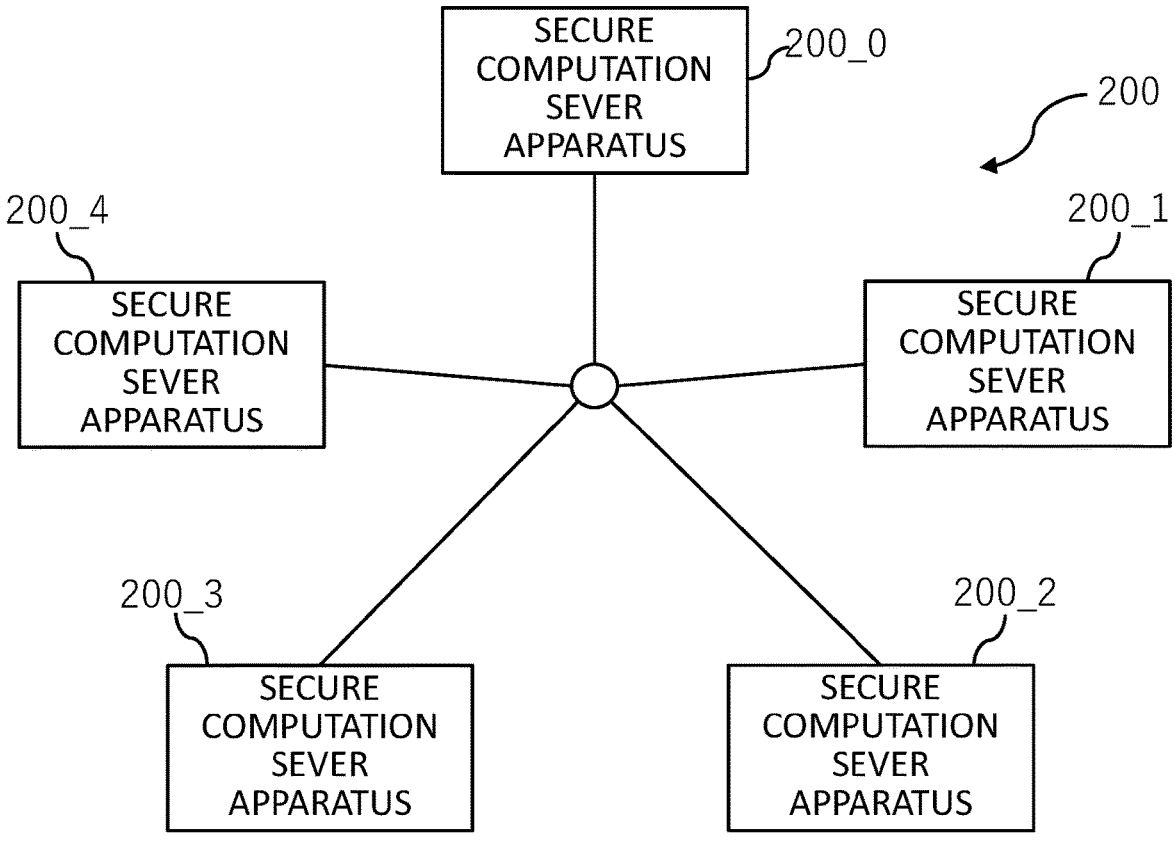

SECURE COMPUTATION SEVER APPARATUS
200_0
200
200_4
SECURE COMPUTATION SEVER APPARATUS
200_1
SECURE COMPUTATION SEVER APPARATUS
200_3
SECURE COMPUTATION SEVER APPARATUS
200_2
SECURE COMPUTATION SEVER APPARATUS

FIG. 5

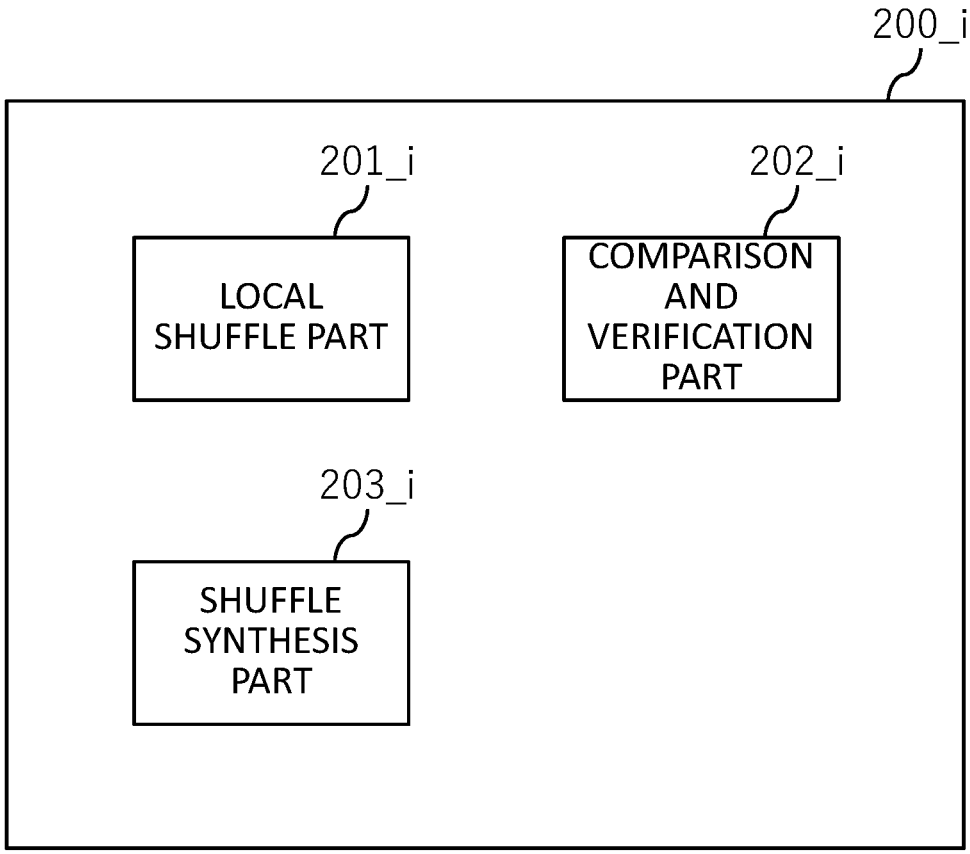
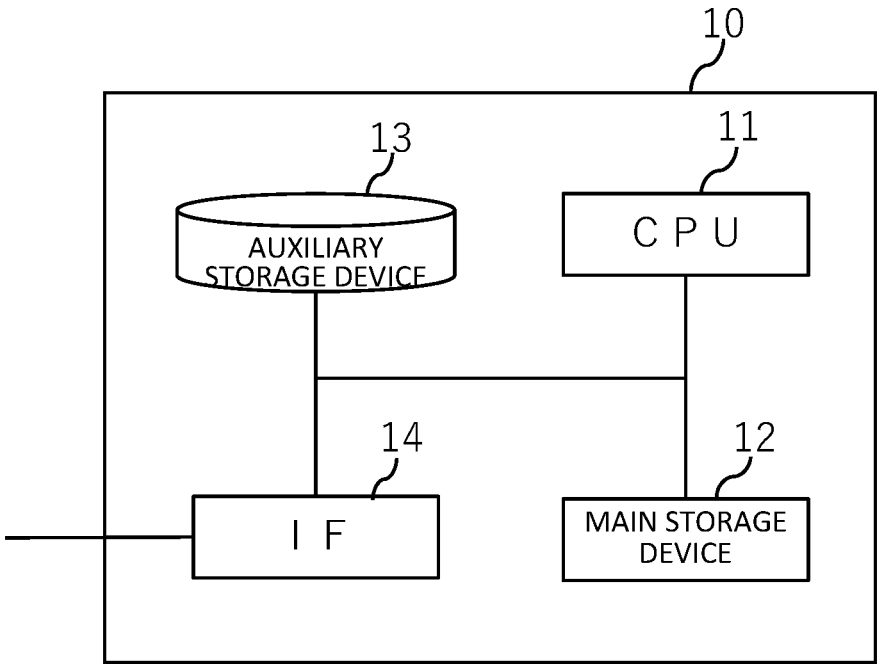

200_i
201_i
LOCAL SHUFFLE PART
202_i
COMPARISON AND VERIFICATION PART
203_i
SHUFFLE SYNTHESIS PART

FIG. 6


10
13
11
AUXILIARY STORAGE DEVICE
C P U
14
12
I F
MAIN STORAGE DEVICE

SECURE COMPUTATION SYSTEM, SECURE COMPUTATION SERVER APPARATUS, SECURE COMPUTATION METHOD, AND SECURE COMPUTATION PROGRAM

TECHNICAL FIELD

[0001] The present invention relates to a secure computation system, a secure computation server apparatus, a secure computation method, and a secure computation program.

BACKGROUND ART

[0002] In recent years, researches and developments on techniques referred to as secure computation are active. Secure computation is one of the techniques for executing predetermined processing while keeping its computation processes and the results thereof secret to third parties. One typical technique used for secure computation is a multiparty computation technique. In this multiparty computation technique, data that needs to be kept secret is distributed to a plurality of servers (secure computation server apparatuses), and each server performs various operations on the data distributed thereto while keeping the data secret. The data distributed to the individual secure computation server apparatuses is called "shares". Hereinafter, unless otherwise stated, the term "secure computation" signifies the multiparty computation technique.

[0003] In the secure computation as described above, computation protocols for specific use are usually implemented in addition to four basic arithmetic operations. One example of these computation protocols for specific use is a shuffle protocol. If the shuffle protocol is used as one of the secure computation protocols, for example, sorting of the shares can be performed efficiently.

[0004] This sorting is a process of comparing the magnitudes of the values, which are kept secret as the shares, with each other and rearranging the shares based on the magnitude comparison result. The magnitude comparison result obtained in the sorting also needs to be kept secret. For example, information about the result of the comparison between the value of the share having a sequence number i and the value of the share having a sequence location j indirectly includes information about the values of the shares having the sequence numbers i and j. If this information is leaked, a security problem is caused. Thus, it is necessary to perform the sorting while keeping secret not only the values of the shares but also the magnitude comparison result. However, performing the computation while keeping the magnitude comparison result secret results in a poor efficiency (a large communication cost).

[0005] If a shuffle is performed before the sorting, the relationship between the sequence numbers and the values of the shares is lost. Thus, there is no need to keep the magnitude comparison result secret. In this way, since there is no need to keep the comparison result secret, the inefficient computation does not need to be performed, and the sorting efficiency is consequently improved. Therefore, implementing the shuffle protocol as one of the secure computation protocols is highly beneficial.

CITATION LIST

Non-Patent Literature

[0006] NPL 1: Byali, M., Chaudhari, H., Patra, A., & Suresh, A. (2020). FLASH: fast and robust framework for privacy-preserving machine learning. Proceedings on Privacy Enhancing Technologies, 2020(2), 459-480.

SUMMARY

Technical Problem

[0007] The disclosure of the above citation list is incorporated herein in its entirety by reference thereto. The following analysis has been made by the present inventor.

[0008] Different techniques that are generally referred to as secure computation achieve different security levels. For example, a case in which one of the participants in a multiparty secure computation is a dishonest person will be considered. In this case, it is possible to adopt a secure computation technique that can detect the presence of the dishonest person and can abort its processes. Alternatively, it is possible to adopt a secure computation technique that can obtain an accurate computation result without aborting its processes even if there is the dishonest person. The latter technique achieves a higher security than the former technique. The secure computation satisfying the latter security is referred to as Guaranteed Output Delivery (GOD), and an example of the secure computation realizing this GOD is known (for example, see NPL 1).

[0009] In addition, regarding the evaluation of the security in the secure computation, not only the advantageous effects of the security that can be achieved, but also pre-conditions have significant implications. A typical pre-condition is use of a random oracle model as a hash function.

[0010] A hash function is a function that responds a unique output to an input, and it is difficult to deduce the input from the output. However, although it is difficult to deduce the input from the output, there is no guarantee that the input cannot be deduced from the output. Thus, the security is evaluated on the assumption that the hash function used does not have vulnerability. The security based on this assumption is called "as being secure in the random oracle model". The security of the secure computation in NPL 1 is "as being secure in the random oracle model".

[0011] In contrast, there is an expression "as being secure in the standard model", as opposed to "as being secure in the random oracle model". That is, although the input could be deduced from the output of the hash function, if this itself does not mean vulnerability of the secure computation, the security is referred to "as being secure in the standard model". Of course, if the same security level is achieved, the security of the standard model is higher than the security of the random oracle model. Thus, when the shuffle protocol is used, too, it is desirable to achieve Guaranteed Output Delivery (GOD) in the standard model.

[0012] The present invention has been made in view of the above problem, and it is an object of the present invention to provide a secure computation system, a secure computation server apparatus, a secure computation method, and a secure computation program that contribute to a bit conversion that achieves Guaranteed Output Delivery (GOD) in the standard model.

Solution to Problem

[0013] According to a first aspect of the present invention, there is provided a secure computation system, which includes five secure computation server apparatuses connected to each other via a network, an individual one of the

secure computation server apparatuses including: a local shuffle part that computes, by using a shared permutation shared by four of the five secure computation server apparatuses, permuted values of a share for a remaining one of the five secure computation server apparatuses and sends the permuted values of the share to the remaining secure computation server apparatus; a comparison and verification part that compares values with each other, which are received from at least three of the four secure computation server apparatuses and which are supposed to be a same value, and adopts the values that are same at least two values as an accurate permutation; and a shuffle synthesis part that synthesizes, regarding five combinations of four secure computation server apparatuses selected from the five secure computation server apparatuses, mini-shuffles, each of which is constructed by using a shared permutation shared by a corresponding combination of four secure computation server apparatuses and a permutation adopted by a corresponding one of the comparison and verification parts.

[0014] According to a second aspect of the present invention, there is provided a secure computation server apparatus, which is one of five secure computation server apparatuses connected to each other via a network, the secure computation server apparatus including: a local shuffle part that computes, by using a shared permutation shared by four of the five secure computation server apparatuses, permuted values of a share for a remaining one of the five secure computation server apparatuses and sends the permuted values of the share to the remaining secure computation server apparatus; a comparison and verification part that compares values with each other, which are received from at least three of the four secure computation server apparatuses and which are supposed to be a same value, and adopts the values that are same at least two values as an accurate permutation; and a shuffle synthesis part that synthesizes, regarding five combinations of four secure computation server apparatuses selected from the five secure computation server apparatuses, mini-shuffles, each of which is constructed by using a shared permutation shared by a corresponding combination of four secure computation server apparatuses and a permutation adopted by a corresponding one of the comparison and verification parts.

[0015] According to a third aspect of the present invention, there is provided a secure computation method, which uses five secure computation server apparatuses connected to each other via a network, the secure computation method including: causing an individual one of the secure computation server apparatuses to compute, by using a shared permutation shared by four of the five secure computation server apparatuses, permuted values of a share for a remaining one of the five secure computation server apparatuses; causing the individual one of the secure computation server apparatuses to send the permuted values of the share to the remaining secure computation server apparatus; causing the individual one of the secure computation server apparatuses to compare values with each other, which are received from at least three of the four secure computation server apparatuses and which are supposed to be a same value; causing the individual one of the secure computation server apparatuses to adopt the values that are same at least two values as an accurate permutation; and causing the individual one of the secure computation server apparatuses to synthesize, regarding five combinations of four secure computation server apparatuses selected from the five secure computation server apparatuses, mini-shuffles, each of which is constructed by using a shared permutation shared by a corresponding combination of four secure computation server apparatuses and a corresponding permutation adopted.

[0016] According to a fourth aspect of the present invention, there is provided a secure computation program, causing at least five secure computation server apparatuses connected to each other via a network to perform a secure computation, the program including: computing, by using a shared permutation shared by four of the five secure computation server apparatuses, permuted values of a share for a remaining one of the five secure computation server apparatuses; sending the permuted values of the share to the remaining secure computation server apparatus; comparing values with each other, which are received from at least three of the four secure computation server apparatuses and which are supposed to be a same value; adopting the values that are same at least two values as an accurate permutation; and synthesizing, regarding five combinations of four secure computation server apparatuses selected from the five secure computation server apparatuses, mini-shuffles, each of which is constructed by using a shared permutation shared by a corresponding combination of four secure computation server apparatuses and a corresponding permutation adopted. The program can be recorded in a computer-readable storage medium. The storage medium may be a non-transient storage medium such as a semiconductor memory, a hard disk, a magnetic recording medium, or an optical recording medium. The present invention can be embodied as a computer program product.

Advantageous Effects of Invention

[0017] According to the individual aspects of the present invention, it is possible to provide a secure computation system, a secure computation server apparatus, a secure computation method, and a secure computation program that contribute to a bit conversion that achieves Guaranteed Output Delivery (GOD) in the standard model.

BRIEF DESCRIPTION OF DRAWINGS

[0018] FIG. **1** is a block diagram illustrating a functional configuration example of a secure computation system according to a first example embodiment.

[0019] FIG. **2** is a block diagram illustrating a functional configuration example of a secure computation server apparatus according to the first example embodiment.

[0020] FIG. **3** is a flowchart illustrating an outline of a procedure of a secure computation method.

[0021] FIG. **4** is a block diagram illustrating a functional configuration example of a secure computation system according to a second example embodiment.

[0022] FIG. **5** is a block diagram illustrating a functional configuration example of a secure computation server apparatus according to the second example embodiment.

[0023] FIG. **6** is a diagram illustrating a hardware configuration example of a secure computation server apparatus.

DESCRIPTION OF EMBODIMENTS

[0024] Hereinafter, example embodiments of the present invention will be described with reference to the accompanying drawings. However, the present invention is not limited to the following example embodiments. In addition,

in the drawings, the same or equivalent elements are denoted by the same reference characters, as necessary. In addition, the drawings are schematic drawings, and therefore, it should be noted that the sizes, ratios, etc. of the individual elements may differ from their actual sizes, ratios, etc. An element in a drawing may have a portion whose size or ratio differs from that of the portion of the element in a different drawing.

First Example Embodiment

[0025] Hereinafter, a secure computation system and secure computation server apparatuses according to a first example embodiment will be described with reference to FIGS. **1** and **2**. The first example embodiment is an example embodiment for describing only a basic concept of the present invention.

[0026] FIG. **1** is a block diagram illustrating a functional configuration example of a secure computation system according to the first example embodiment. As illustrated in FIG. **1**, a secure computation system **100** according to the first example embodiment includes a first secure computation server apparatus **100_0**, a second secure computation server apparatus **100_1**, a third secure computation server apparatus **100_2**, a fourth secure computation server apparatus **100_3**, and a fifth secure computation server apparatus **100_4**. The first secure computation server apparatus **100_0**, the second secure computation server apparatus **100_1**, the third secure computation server apparatus **100_2**, the fourth secure computation server apparatus **100_3**, and the fifth secure computation server apparatus **100_4** are connected to each other via a network such that these apparatuses can communicate with each other.

[0027] In the secure computation system **100** including the first to fifth secure computation server apparatuses **100_*i*** (i=0, 1, 2, 3, 4), it is possible to compute target shares from a value inputted to any one of the first to fifth secure computation server apparatuses **100_*i*** (i=0, 1, 2, 3, 4) while keeping the input value and the values acquired in the computation processes secret, and it is possible to dispersedly store the computation results in the first to fifth secure computation server apparatuses **100_*i*** (i=0, 1, 2, 3, 4).

[0028] In addition, in the secure computation system **100** including the first to fifth secure computation server apparatuses **100_*i*** (i=0, 1, 2, 3, 4), it is possible to compute target shares from the shares dispersedly stored in the first to fifth secure computation server apparatuses **100_*i*** (i=0, 1, 2, 3, 4) while keeping the values in the computation processes secret, and it is possible to dispersedly store the computation results in the first to fifth secure computation server apparatuses **100_*i*** (i=0, 1, 2, 3, 4).

[0029] The shares of the computation results may be reconstructed by causing the first to fifth secure computation server apparatuses **100_0** to **100_4** to exchange their shares with each other. Alternatively, the shares may be decoded by transmitting the shares to an external apparatus other than the first to fifth secure computation server apparatuses **100_0** to **100_4**.

[0030] In addition, in the secure computation system **100** including the first to fifth secure computation server apparatuses **100_*i*** (i=0, 1, 2, 3, 4), even when one of the first to fifth secure computation server apparatuses **100_*i*** (i=0, 1, 2, 3, 4) is operated by a dishonest person, it is possible to continue an accurate secure computation without stopping the processes.

[0031] For example, the following construction may be adopted as the construction of the shares that enables continuation of an accurate secure computation without stopping the processes even when one of the first to fifth secure computation server apparatuses **100_*i*** (i=0, 1, 2, 3, 4) is operated by a dishonest person as described above.

[0032] Shares of an element x of a residue class ring Z_n of modulo n, that is, $x \in Z_n$, on the residue class ring Z_n are defined as follows (the shares may be referred to as arithmetic shares, as necessary). Note that $n=2^m$, where m is an integer of 2 or more. That is, a residue class ring Z_2 of modulo 2 is distinguished from the residue class ring Z_n of modulo n.

[0033] An element x of the residue class ring Z_n of modulo n, that is, $x \in Z_n$, is decomposed to satisfy the following relationship:

[0034] $x=x_0+x_1+x_2+x_3+x_4 \bmod n$

[0035] $[x]_i$ dispersedly held by the individual participants P_i, (i=0, 1, 2, 3, 4) is defined as follows.

[0036] $[x]_i=(x_i, x_{i+1}, x_{i+2}, x_{i+3}$, note that $x_{4+1}=x_0$

[0037] Shares of an element x of the residue class ring Z_2 of modulo 2, that is, $x \in Z_2$, on the residue class ring Z_2 (the shares may be referred to as logic shares, as necessary) are defined in the same way as the above shares on the residue class ring Z_n where n=2. However, a different notation $[x]^B$ is used to distinguish the residue class ring Z_2 of modulo 2 from the residue class ring Z_n of modulo n. That is, the shares are specifically defined as follows.

[0038] An element x of the residue class ring Z_2 of modulo 2, that is, $x \in Z_2$, is decomposed as follows. In Equation 1, "+" inside a circle represents an exclusive-or.

$$x=x_0 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4 \bmod 2 \quad \text{[Equation 1]}$$

[0039] $[x]^B{}_i$ dispersedly held by the individual participants P_i, (i=0, 1, 2, 3, 4) is defined as follows.

[0040] $[x]^B{}_i=(x_i, x_{i+1}, x_{i+2}, x_{i+3})$, note that $x_{4+1}=x_0$

[0041] If these shares $[x]_0$, $[x]_1$, $[x]_2$, $[x]_3$, and $[x]_4$ held by the individual participants P_i, (i=0, 1, 2, 3, 4) are determined as described above, the individual participants P_i, (i=0, 1, 2, 3, 4) cannot reconstruct x from their shares $[x]_0$, $[x]_1$, $[x]_2$, $[x]_3$, and $[x]_4$ held thereby. However, it is possible to realize secret sharing in which x can be reconstructed if the shares held by at least two of the participants P_i (i=0, 1, 2, 3, 4) are combined. This secret sharing scheme is referred to as a 2-out-of-5 Replicated Secret Sharing Scheme.

[0042] Herein, a shuffle is used when a share sequence constructed as described above is dispersedly held by the individual participant. That is, an individual component x_j of a sequence having a sequence length M is decomposed into shares as follows, and the shares are dispersedly held by the individual participants P_i (i=0, 1, 2, 3, 4).

$$\vec{x}=(x_0, \ldots, x_1, \ldots, x_{M-1})$$

$$x_j=x_{j,0}+x_{j,1}+x_{j,2}+x_{j,3}+x_{j,4} \bmod L$$

$$\vec{x_i}=(x_{0,i}, \ldots, x_{M-1,i})(i=0,1,2,3,4) \quad \text{[Equations 2]}$$

[0043] In this secure computation based on the secret sharing scheme, not only when x is reconstructed but also when the locations in a sequence are shuffled, there is a situation in which the individual participants directly or indirectly receive the values of the sub-shares not held thereby from other participants. This is because it is necessary to shuffle the locations in a sequence while maintaining the values indicated by the shares held in a secret sharing

manner by the individual participants. In addition, it is necessary that the values indicated by the shares be maintained even after the locations in the sequence are changed without letting the individual participants know the values indicated by their shares held thereby.

[0044] Thus, when a bit conversion is performed, too, there is a situation in which the individual participants also directly or indirectly receive the values of the sub-shares not held thereby from other participants. In this situation, if one of the other participants is a dishonest person, a participant could receive a different value instead of a value that the participant is originally supposed to receive. If this happens, the secure computation is performed based on an erroneous value, resulting in an erroneous result. In some cases, the computation itself cannot be performed properly.

[0045] To solve this problem, in the secure computation system **100** according to the present example embodiment, as illustrated in FIG. **2**, an individual one of the first to fifth secure computation server apparatuses **100**_*i* (i=0, 1, 2, 3, 4) includes a local shuffle part **101**_*i*, a comparison and verification part **102**_*i*, and a shuffle synthesis part **103**_*i*. FIG. **2** is a block diagram illustrating a functional configuration example of a secure computation server apparatus according to the first example embodiment.

[0046] The local shuffle part **101**_*i* computes, by using a shared permutation shared by four of the first to fifth secure computation server apparatuses **100**_*i* (i=0, 1, 2, 3, 4), permuted values of a share for a remaining one of the first to fifth secure computation server apparatuses **100**_*i* (i=0, 1, 2, 3, 4) and sends the permuted values of the share to the remaining secure computation server apparatus. The comparison and verification part **102**_*i* compares the permuted values of the share with each other, which are received from at least three of the four secure computation server apparatuses and which are supposed to be the same value, and adopts the values that are same at least two values as an accurate permutation. The shuffle synthesis part **103**_*i* synthesizes, regarding five combinations of four secure computation server apparatuses selected from the five secure computation server apparatuses **100**_*i* (i=0, 1, 2, 3, 4), mini-shuffles, each of which is constructed by using a shared permutation shared by a corresponding combination of four secure computation server apparatuses and a permutation adopted by a corresponding one of the comparison and verification parts **102**_*i*.

[0047] As described above, in the secure computation system **100** according to the present example embodiment, an individual one of the four of the first to fifth secure computation server apparatuses **100**_*i* (i=0, 1, 2, 3, 4) permutes the locations of its sub-shares by using a permutation completed thereby and also performs a permutation for the remaining one of the secure computation server apparatuses. This remaining secure computation server apparatus receives the sub-shares permuted by the four secure computation server apparatuses from these four secure computation server apparatuses, compares the permuted values of the share, which are received from at least three of the four secure computation server apparatuses and which are supposed to be the same value, and adopts the values that are same at least two values as an accurate permutation. As a result, mini-shuffles are constructed in the first to fifth secure computation server apparatuses **100**_*i* (i=0, 1, 2, 3, 4).

[0048] It should be noted here that, since each of the four secure computation server apparatuses completes a permutation by itself, each computation server apparatus knows how the locations in the share sequence have been permuted, and that because the remaining one of the secure computation server apparatuses receives a permuted result, this remaining secure computation server apparatus does not know how the locations in the share sequence have been permuted. Thus, regarding the five combinations of four secure computation server apparatuses selected from the first to fifth secure computation server apparatuses **100**_*i* (i=0, 1, 2, 3, 4), if all the mini-shuffles constructed as described above are synthesized, a permutation (shuffle) that cannot be traced by any one of the first to fifth secure computation server apparatuses **100**_*i* (i=0, 1, 2, 3, 4) can be constructed.

[0049] Next, a secure computation method according to the present example embodiment will be described. FIG. **3** is a flowchart illustrating an outline of a procedure of the secure computation method.

[0050] As illustrated in FIG. **3**, the secure computation method according to the present example embodiment includes a local shuffle step (S**11**), a comparison and verification step (S**12**), and a shuffle synthesis step (S**13**). In the local shuffle step (S**11**), an individual secure computation server apparatus computes, by using a shared permutation shared by four of the first to fifth secure computation server apparatuses **100**_*i* (i=0, 1, 2, 3, 4), permuted values of a share for a remaining one of the first to fifth secure computation server apparatuses **100**_*i* (i=0, 1, 2, 3, 4) and sends the permuted values of the share to the remaining secure computation server apparatus. Next, in the comparison and verification step (S**12**), the individual secure computation server apparatus compares the permuted values of the share with each other, which are received from at least three of the four secure computation server apparatuses and which are supposed to be the same value, and adopts the values that are same at least two values as an accurate permutation. Finally, in the shuffle synthesis step (**13**), the individual secure computation server apparatus synthesizes, regarding five combinations of four secure computation server apparatuses selected from the five secure computation server apparatuses **100**_*i* (i=0, 1, 2, 3, 4), mini-shuffles, each of which is constructed by using a shared permutation shared by a corresponding combination of four secure computation server apparatuses and a permutation adopted by a corresponding one of the comparison and verification parts **102**_*i*.

[0051] As described above, in the secure computation system **100** and the secure computation method according to the present example embodiment, a participant receives received values, which are received from at least three of the other participants and which are supposed to be the same value, and adopts the received values that are same at least two received values as an accurate value. In this way, even if one of the other participants is a dishonest person, the participants can determine an accurate value. That is, even if there is a dishonest person, it is possible to realize Guaranteed Output Delivery (GOD) that can acquire an accurate computation without stopping the processes. In addition, because no hash function is used in the above processes, Guaranteed Output Delivery (GOD) is realized in a normal model.

[0052] The first example embodiment described above is an example embodiment for describing only a basic concept of the present invention. A second example embodiment

described below is a practical example embodiment to which the above-described concept is applied.

Second Example Embodiment

[0053] Hereinafter, a secure computation system and secure computation server apparatuses according to a second example embodiment will be described with reference to FIGS. **4** and **5**.

[0054] FIG. **4** is a block diagram illustrating a functional configuration example of a secure computation system according to the second example embodiment. As illustrated in FIG. **4**, a secure computation system **200** according to the second example embodiment includes a first secure computation server apparatus **200_0**, a second secure computation server apparatus **200_1**, a third secure computation server apparatus **200_2**, a fourth secure computation server apparatus **200_3**, and a fifth secure computation server apparatus **200_4**. The first secure computation server apparatus **200_0**, the second secure computation server apparatus **200_1**, the third secure computation server apparatus **200_2**, the fourth secure computation server apparatus **200_3**, and the fifth secure computation server apparatus **200_4** are connected to each other via a network such that these apparatuses can communicate with each other.

[0055] In the secure computation system **200** including the first to fifth secure computation server apparatuses **200_*i*** (i=0, 1, 2, 3, 4), it is possible to compute target shares from a value inputted to any one of the first to fifth secure computation server apparatuses **200_*i*** (i=0, 1, 2, 3, 4) while keeping the input value and the values acquired in the computation processes secret, and it is possible to dispersedly store the computation results in the first to fifth secure computation server apparatuses **200_*i*** (i=0, 1, 2, 3, 4).

[0056] In addition, in the secure computation system **200** including the first to fifth secure computation server apparatuses **200_*i*** (i=0, 1, 2, 3, 4), even when one of the first to fifth secure computation server apparatuses **200_*i*** (i=0, 1, 2, 3, 4) is operated by a dishonest person, it is possible to continue an accurate secure computation without stopping the processes.

[0057] FIG. **5** is a block diagram illustrating a functional configuration example of a secure computation server apparatus according to the second example embodiment. As illustrated in FIG. **5**, in the secure computation system **200** according to the present example embodiment, an individual one of the first to fifth secure computation server apparatuses **200_*i*** (i=0, 1, 2, 3, 4) includes a local shuffle part **201_*i***, a comparison and verification part **202_*i***, and a shuffle synthesis part **203_*i***.

[0058] The local shuffle part **201_*i*** computes, by using a shared permutation shared by four of the first to fifth secure computation server apparatuses **200_*i*** (i=0, 1, 2, 3, 4), permuted values of a share for a remaining one of the first to fifth secure computation server apparatuses **200_*i*** (i=0, 1, 2, 3, 4) and sends the permuted values of the share to the remaining secure computation server apparatus. The comparison and verification part **202_*i*** compares the permuted values of the share with each other, which are received from at least three of the four secure computation server apparatuses and which are supposed to be the same value, and adopts the values that are same at least two values as an accurate permutation. The shuffle synthesis part **203_*i*** synthesizes, regarding five combinations of four secure computation server apparatuses selected from the five secure computation server apparatuses **200_*i*** (i=0, 1, 2, 3, 4), mini-shuffles, each of which is constructed by using a shared permutation shared by a corresponding combination of four secure computation server apparatuses and a permutation adopted by a corresponding one of the comparison and verification parts **202_*i***.

[0059] Hereinafter, building blocks used for execution of the bit conversion according to the present example embodiment will be described.

[Generation of Pseudo Random Numbers and Sharing of Seeds] A pseudo-random function F_n, seeds, and an identifier have a relationship as follows. The pseudo-random function F_n, is a binary operation defined with a security parameter x.

$$F_n:\{0,1\}K^k\times\{0,1\}^k\rightarrow\{0,1\}^n$$

Seeds $\text{seed}_i\in\{0,1\}^k$ ((i=0, 1, 2, 3, 4) are values appropriately shared by the individual secure computation server apparatuses **200_*i***, and an identifier vid $\in\{0,1\}^k$ is a public value such as a counter. The pseudo-random function F_n determinably generates pseudo random numbers by using the seeds and the identifier as its inputs.

[0060] Regarding the five seeds $\text{seed}_i\in\{0,1\}^k$ ((i=0, 1, 2, 3, 4), an individual one of the secure computation server apparatuses **200_*i*** holds (seed_i, seed_{i+1}, seed_{i+2}, seed_{i+3}). Note that $\text{seed}_{4+1}=\text{seed}_0$. That is, an individual one of the secure computation server apparatuses **200_*i*** holds the seeds seed, other than the seed seed_{i+4}. For example, the sharing of these seeds can be appropriately set by an administrator or the like as a presetting of the secure computation server apparatuses **200_*i***.

[0061] [Creation of Mask]

[0062] Next, a pseudo random number (Correlated Randomness) that is seen as a random number by the participant P_{i+4} and cannot be removed and that can be determinably computed by the other participants P_i, P_{i+1}, P_{i+2}, and P_{i+3} is created, and this pseudo random number will be used as a mask when the permuted shares, which will be described below, are sent.

[0063] First, since the participant P_{i+4} does not hold the seed seed_{i+3}, if the seed seed_{i+3} is used as an input of the pseudo-random function F_n, the following pseudo random number satisfies the above condition. That is, although the following r_k is seen as a random number by the participant P_{i+4} and cannot be removed, the following r_k can be determinably computed by the other participants P_i, P_{i+1}, P_{i+2}, and P_{i+3}. $r_k=F_n(\text{vid}_k, \text{seed}_{i+3})-F_n(\text{vid}_{k+1}, \text{seed}_{i+3})$ mod n

[0064] In addition, by changing the index k in the identifier vid_k from k=0 to k=4, five pseudo random numbers r_k can be created. A set of these pseudo random numbers r_k is defined as follows. Whether the following pseudo random numbers r_0, r_1, r_2, r_3, and r_4 determined as follows satisfy $r_0+r_1+r_2+r_3+r_4=0$ can be easily determined.

$$(r_0,r_1,r_2,r_3,r_4)=CR(i+4,\{vid_k\}^4_{k=0},\text{seed}_{i+3})$$

[0065] Although the pseudo random numbers r_0, r_1, r_2, r_3, and r_4 created as described above are seen as random numbers by the participant P_{i+4} and cannot be removed, these pseudo random numbers can be determinably computed by the other participants P_i, P_{i+1}, P_{i+2}, and P_{i+3}. However, although the pseudo random numbers r_0, r_1, r_2, r_3, and r_4 cannot be removed by the participant P_{i+4}, if all the pseudo random numbers r_0, r_1, r_2, r_3, and r_4 are collected, because the sum is 0, the pseudo random numbers r_0, r_1, r_2, r_3, and r_4 can be removed by the participant P_{i+4}.

[0066] [Construction of Mini-Shuffles]

[0067] The construction of a mini-shuffle corresponds to the local shuffle step (**S11**) and the comparison and verification step (**S12**) described in the above first example embodiment.

[0068] The following description will be made on an example in which, among the participants P_i (i=0, 1, 2, 3, 4), each of the participants P_i (i=1, 2, 3, 4) computes a permutation of the participant P_0 for the participant P_0 by using a permutation shared thereby and sends the computed permutation to the participant P_0.

[0069] A permutation $\sigma_0 \in S_M$ shared by the participants P_i (i=1, 2, 3, 4) is constructed as follows. As described above, the participants P_i, (i=0, 1, 2, 3, 4) hold the seeds ($seed_i$, $seed_{i+1}$, $seed_{i+2}$, $seed_{i+3}$). In other words, each participant P_i (i=0, 1, 2, 3, 4) does not hold the seed $seed_{i+4}$. That is, while the participant P_0 does not hold the seed $seed_4$, the other participants P_i (i=1, 2, 3, 4) hold the seed $seed_4$ Thus, the permutation $\sigma_0 \in S_M$ shared by the participants P_i (i=1, 2, 3, 4) is constituted by using a pseudo random number generated by using the seed $seed_4$. In this way, although the permutation $\sigma_0 \in S_M$ is traceable by the participants P_i (i=1, 2, 3, 4), the permutation (To $\in S_M$ is not traceable by the participant P_0.

[0070] Next, by using the permutation $\sigma_0 \in S_M$ constructed as described above and the pseudo random number r_k, each of the participants P_i (i=1, 2, 3, 4) computes the permutation of the participant P_0 for the participant P_0 and sends the computed permutation to the participant P_0 as follows. It should be noted here that, from the method of constructing the shares in secret sharing, there are shares shared by the participants P_i (i=1, 2, 3, 4) and the participant P_0 and that the participants P_i (i=1, 2, 3, 4) can compute the permutation of the participant P_0 for the participant P_0.

$$\{P_2,P_3,P_4\}: \vec{r_0}+\sigma_0(\vec{x_0}) \quad \text{[Equations 3]}$$

$$\{P_1,P_3,P_4\}: \vec{r_1}+\sigma_0(\vec{x_1})$$

$$\{P_1,P_2,P_4\}: \vec{r_2}+\sigma_0(\vec{x_2})$$

$$\{P_1,P_2,P_3\}: \vec{r_3}+\sigma_0(\vec{x_3})$$

$$x_i=(x_{0,i}, \ldots, x_{M-1,i})(i=0,1,2,3)$$

$$x_j=x_{j,0}+x_{j,1}+x_{j,2}+x_{j,3}+x_{j,4} \bmod L$$

$$r_i=(r_{i,0}, \ldots, r_{i,j}, \ldots, r_{i,m-1})$$

$$r_{0,j}+r_{1,j}+r_{2,j}+r_{3,j}+r_{4,j}=0 \bmod L(j=0, \ldots, m-1)$$

[0071] In the above transmission, note that the participants P_2, P_3, P_4 send the same value to the participant P_0, the participants P_1, P_3, P_4 send the same value to the participant P_0, the participants P_1, P_2, P_4 send the same value to the participant P_0, the participants P_1, P_2, P_3 send the same value to the participant P_0. From this nature, the participant P_0 can compare the values of the permuted shares, which are received from three participants and which are supposed to be the same value, and can adopt the received values that are same at least two received values as an accurate permutation. That is, even if one of the other participants is a dishonest person, the participants can determine an accurate value. That is, even if there is a dishonest person, it is possible to realize Guaranteed Output Delivery (GOD) that can acquire an accurate computation without stopping the processes.

[0072] In addition, in the above transmission, by using a hash function shared by the participants P_i (i=1, 2, 3, 4) and transmitting the hash value as follows, the communication amount can be reduced. First, one of the three participants converts a value by using the hash function and sends the obtained hash value to the participant P_0. The other two participants send the value, not a hash value, to the participant P_0 without change. Upon receiving the value, the participant P_0 converts the value into a hash value by using the hash function. Next, the participant P_0 compares the hash values with each other. If at least two of the hash values are the same value, the participant P_0 adopts this value as an accurate value.

[0073] By performing the process as described above, a share corresponding to the permutation $\sigma_0 \in S_M$ that can be computed by the participants P_i (i=1, 2, 3, 4) and that cannot be computed by the participant P_0 can be constituted as follows.

$$P_i:[\sigma_0(\vec{x})]_{1,i}=(\vec{r_1}+\sigma_0(\vec{x_1}),\vec{r_{i+1}}+\sigma_0(\vec{x_{i+1}}),\vec{r_{i+2}}+\sigma_0(\vec{x_{i+2}}),\vec{r_{i+3}}\sigma_0(\vec{x_{i+3}}))$$

$$\vec{x_i}=(x_{0,i}, \ldots, x_{M-1,i})(i=0,1,2,3)$$

$$x_j=x_{j,0}+x_{j,1}+x_{j,2}+x_{j,3}+x_{j,4} \bmod L$$

$$r_1=(r_{i,0}, \ldots, r_{i,j}, \ldots, r_{i,m-1})$$

$$r_{0,j}+r_{1,j}+r_{2,j}+r_{3,j}+r_{4,j}=0 \bmod L(j=0, \ldots, m-1) \quad \text{[Equations 4]}$$

[0074] [Synthesis of Mini-Shuffles]

[0075] In the above construction of mini-shuffles, of all the participants P_i (i=0, 1, 2, 3, 4), each of the participants P_i (i=1, 2, 3, 4) computes the permutation of the participant P_0 for the participant P_0 by using a permutation shared by the participants P_i (i=1, 2, 3, 4) and sends the computed permutation to the participant P_0. However, alternatively, a combination of four participants P_i and one participant P_0 may be changed. In this way, five mini-shuffles can be constructed. Each of these five mini-shuffles represents a permutation that is not traceable by one of the participants P_i, (i=0, 1, 2, 3, 4).

[0076] Thus, by synthesizing all the permutations σ_i (i=0, 1, 2, 3, 4)$\in S_M$, each of which is not traceable by one of the participants P_i, (i=0, 1, 2, 3, 4), a permutation σ (shuffle) that is not traceable by any one of the participants P_i (i=0, 1, 2, 3, 4) can be constructed.

$$\sigma=\sigma_0{}^\circ\sigma_1{}^\circ\sigma_2{}^\circ\ \sigma_3{}^\circ\sigma_4 \quad \text{[Equation 5]}$$

[0077] Regarding the communication cost of the shuffle constructed as described above, the number of communication rounds is 5, and the communication amount is 40 nm bits. Regarding a single mini-shuffle, the number of communication rounds is 1, and the communication amount is 8 nm bits. Because five mini-shuffles are performed, the above communication cost is achieved.

Hardware Configuration Example

[0078] FIG. **6** is a diagram illustrating a hardware configuration example of a secure computation server apparatus. That is, the hardware configuration example illustrated in FIG. **6** is a hardware configuration example of any one of

the secure computation server apparatuses **100**_*i* and **200**_*i* (i=0, 1, 2, 3, 4). An information processing apparatus (a computer) that adopts the hardware configuration illustrated in FIG. **6** can realize the individual functions of any one of the secure computation server apparatuses **100**_*i* and **200**_*i* (i=0, 1, 2, 3, 4) by executing the corresponding one of the above secure computation methods as a program.

[0079] The hardware configuration example illustrated in FIG. **6** is an example of the hardware configuration that realizes the individual functions of any one of the secure computation server apparatuses **100**_*i* and **200**_*i* (i=0, 1, 2, 3, 4), and does not limit the hardware configuration of any one of the secure computation server apparatuses **100**_*i* and **200**_*i* (i=0, 1, 2, 3, 4). The secure computation server apparatuses **100**_*i* and **200**_*i* (i=0, 1, 2, 3, 4) may include hardware not illustrated in FIG. **6**.

[0080] As illustrated in FIG. **6**, a hardware configuration **10** that can be adopted by any one of the secure computation server apparatuses **100**_*i* and **200**_*i* (i=0, 1, 2, 3, 4) includes, for example, a CPU (Central Processing Unit) **11**, a main storage device **12**, an auxiliary storage device **13**, and an IF (Interface) part **14**, which are connected to each other via an internal bus.

[0081] The CPU **11** executes various commands included in the secure computation program executed by the corresponding one of the secure computation server apparatuses **100**_*i* and **200**_*i* (i=0, 1, 2, 3, 4). The main storage device **12** is, for example, a RAM (Random Access Memory) and temporarily stores various kinds of programs such as the secure computation program executed by the corresponding one of the secure computation server apparatuses **100**_*i* and **200**_*i* (i=0, 1, 2, 3, 4) so that the CPU **11** can execute the programs.

[0082] The auxiliary storage device **13** is, for example, an HDD (Hard Disk Drive) and can store, in the mid-to-long term, various kinds of programs such as the secure computation program executed by the corresponding one of the secure computation server apparatuses **100**_*i* and **200**_*i* (i=0, 1, 2, 3, 4). These various kinds of programs such as the secure computation program can be recorded in a non-transitory computer-readable storage medium and can be provided as a program product. The auxiliary storage device **13** can be used to store, in the mid-to-long term, various kinds of programs such as the secure computation program recorded in a non-transitory computer-readable storage medium. The IF part **14** provides an interface regarding the input and output among the corresponding secure computation server apparatuses **100**_*i* and **200**_*i* (i=0, 1, 2, 3, 4).

[0083] An information processing apparatus that adopts the hardware configuration **10** as described above realizes the functions of any one of the secure computation server apparatuses **100**_*i* and **200**_*i* (i=0, 1, 2, 3, 4) by executing the corresponding one of the above-described secure computation methods as a program.

[0084] The above example embodiments can partially or entirely be described, but not limited to, as the following notes.

[Note 1]

[0085] A secure computation system, which includes five secure computation server apparatuses connected to each other via a network, an individual one of the secure computation server apparatuses including:

[0086] a local shuffle part that computes, by using a shared permutation shared by four of the five secure computation server apparatuses, permuted values of a share for a remaining one of the five secure computation server apparatuses and sends the permuted values of the share to the remaining secure computation server apparatus;

[0087] a comparison and verification part that compares values with each other, which are received from at least three of the four secure computation server apparatuses and which are supposed to be a same value, and adopts the values that are same at least two values as an accurate permutation; and

[0088] a shuffle synthesis part that synthesizes, regarding five combinations of four secure computation server apparatuses selected from the five secure computation server apparatuses, mini-shuffles, each of which is constructed by using a shared permutation shared by a corresponding combination of four secure computation server apparatuses and a permutation adopted by a corresponding one of the comparison and verification parts.

[Note 2]

[0089] The secure computation system according to note 1; wherein the mini-shuffles are each determinably generated by using seeds held by the four secure computation server apparatuses and an identifier held by the five secure computation server apparatuses as input and are each masked by pseudo random numbers whose total is zero regarding the five secure computation server apparatuses.

[Note 3]

[0090] The secure computation system according to note 1 or 2; wherein the shared permutation is determinably generated by using a seed shared by the four secure computation server apparatuses as input.

[Note 4]

[0091] The secure computation system according to any one of notes 1 to 3; wherein the comparison and verification part determines that the received values are each an accurate value by determining that hash values of the received values are same.

[Note 5]

[0092] A secure computation server apparatus, which is one of five secure computation server apparatuses connected to each other via a network, the secure computation server apparatus including:

[0093] a local shuffle part that computes, by using a shared permutation shared by four of the five secure computation server apparatuses, permuted values of a share for a remaining one of the five secure computation server apparatuses and sends the permuted values of the share to the remaining secure computation server apparatus;

[0094] a comparison and verification part that compares values with each other, which are received from at least three of the four secure computation server apparatuses and which are supposed to be a same value, and adopts the values that are same at least two values as an accurate permutation; and

[0095] a shuffle synthesis part that synthesizes, regarding five combinations of four secure computation server apparatuses selected from the five secure computation server apparatuses, mini-shuffles, each of which is constructed by using a shared permutation shared by a corresponding combination of four secure computation server apparatuses and a permutation adopted by a corresponding one of the comparison and verification parts.

[Note 6]

[0096] A secure computation method, which uses five secure computation server apparatuses connected to each other via a network, the secure computation method including:

[0097] causing an individual one of the secure computation server apparatuses to compute, by using a shared permutation shared by four of the five secure computation server apparatuses, permuted values of a share for a remaining one of the five secure computation server apparatuses;

[0098] causing the individual one of the secure computation server apparatuses to send the permuted values of the share to the remaining secure computation server apparatus;

[0099] causing the individual one of the secure computation server apparatuses to compare values with each other, which are received from at least three of the four secure computation server apparatuses and which are supposed to be a same value;

[0100] causing the individual one of the secure computation server apparatuses to adopt the values that are same at least two values as an accurate permutation; and

[0101] causing the individual one of the secure computation server apparatuses to synthesize, regarding five combinations of four secure computation server apparatuses selected from the five secure computation server apparatuses, mini-shuffles, each of which is constructed by using a shared permutation shared by a corresponding combination of four secure computation server apparatuses and a corresponding permutation adopted.

[Note 7]

[0102] The secure computation method according to note 6; wherein the mini-shuffles are each determinably generated by using seeds held by the four secure computation server apparatuses and an identifier held by the five secure computation server apparatuses as input and are each masked by pseudo random numbers whose total is zero regarding the five secure computation server apparatuses.

[Note 8]

[0103] The secure computation method according to note 6 or 7; wherein the shared permutation is determinably generated by using a seed shared by the four secure computation server apparatuses as input.

[Note 9]

[0104] The secure computation method according to any one of notes 6 to 8; wherein it is determined that the permuted values of the share are each an accurate value by determining that hash values of the received values are same.

[Note 10]

[0105] A secure computation program, causing at least five secure computation server apparatuses connected to each other via a network to perform a secure computation, the program including:

[0106] computing, by using a shared permutation shared by four of the five secure computation server apparatuses, permuted values of a share for a remaining one of the five secure computation server apparatuses;

[0107] sending the permuted values of the share to the remaining secure computation server apparatus;

[0108] comparing values with each other, which are received from at least three of the four secure computation server apparatuses and which are supposed to be a same value;

[0109] adopting the values that are same at least two values as an accurate permutation; and

[0110] synthesizing, regarding five combinations of four secure computation server apparatuses selected from the five secure computation server apparatuses, mini-shuffles, each of which is constructed by using a shared permutation shared by a corresponding combination of four secure computation server apparatuses and a corresponding permutation adopted.

[0111] The disclosure of the above NPL is incorporated herein by reference thereto. Modifications and adjustments of the example embodiments or examples are possible within the scope of the overall disclosure (including the claims) of the present invention and based on the basic technical concept of the present invention. Various combinations or selections (including partial deletion) of various disclosed elements (including the elements in each of the claims, example embodiments, examples, drawings, etc.) are possible within the scope of the disclosure of the present invention. That is, the present invention of course includes various variations and modifications that could be made by those skilled in the art according to the overall disclosure including the claims and the technical concept. The description discloses numerical value ranges. However, even if the description does not particularly disclose arbitrary numerical values or small ranges included in the ranges, these values and ranges should be deemed to have been specifically disclosed. In addition, as needed and based on the gist of the present invention, partial or entire use of the individual disclosed matters in the above literatures that have been referred to in combination with what is disclosed in the present application should be deemed to be included in what is disclosed in the present application, as a part of the disclosure of the present invention.

REFERENCE SIGNS LIST

[0112] **100**, **200** secure computation system

[0113] **100**_*i*, **200**_*i* secure computation server apparatus

[0114] **101**_*i*, **201**_*i* local shuffle part

[0115] **102**_*i*, **202**_*i* comparison and verification part

[0116] **103**_*i*, **203**_*i* shuffle synthesis part

[0117] **10** hardware configuration

[0118] **11** CPU (Central Processing Unit)

[0119] **12** main storage device

[0120] **13** auxiliary storage device

[0121] **14** IF (Interface) part

What is claimed is:

1. A secure computation system, which includes five secure computation server apparatuses connected to each other via a network, an individual one of the secure computation server apparatuses comprising:

a local shuffle part that computes, by using a shared permutation shared by four of the five secure computation server apparatuses, permuted values of a share for a remaining one of the five secure computation server apparatuses and sends the permuted values of the share to the remaining secure computation server apparatus;

a comparison and verification part that compares values with each other, which are received from at least three of the four secure computation server apparatuses and which are supposed to be a same value, and adopts the values that are same at least two values as an accurate permutation; and

a shuffle synthesis part that synthesizes, regarding five combinations of four secure computation server apparatuses selected from the five secure computation server apparatuses, mini-shuffles, each of which is constructed by using a shared permutation shared by a corresponding combination of four secure computation server apparatuses and a permutation adopted by a corresponding one of the comparison and verification parts.

2. The secure computation system according to claim **1**; wherein the mini-shuffles are each determinably generated by using seeds held by the four secure computation server apparatuses and an identifier held by the five secure computation server apparatuses as input and are each masked by pseudo random numbers whose total is zero regarding the five secure computation server apparatuses.

3. The secure computation system according to claim **1**; wherein the shared permutation is determinably generated by using a seed shared by the four secure computation server apparatuses as input.

4. The secure computation system according to claim **1**; wherein the comparison and verification part determines that the received values are each an accurate value by determining that hash values of the received values are same.

5. A secure computation server apparatus, which is one of five secure computation server apparatuses connected to each other via a network, the secure computation server apparatus comprising:

a local shuffle part that computes, by using a shared permutation shared by four of the five secure computation server apparatuses, permuted values of a share for a remaining one of the five secure computation server apparatuses and sends the permuted values of the share to the remaining secure computation server apparatus;

a comparison and verification part that compares values with each other, which are received from at least three of the four secure computation server apparatuses and which are supposed to be a same value, and adopts the values that are same at least two values as an accurate permutation; and

a shuffle synthesis part that synthesizes, regarding five combinations of four secure computation server apparatuses selected from the five secure computation server apparatuses, mini-shuffles, each of which is constructed by using a shared permutation shared by a corresponding combination of four secure computation server apparatuses and a permutation adopted by a corresponding one of the comparison and verification parts.

6. A secure computation method, which uses five secure computation server apparatuses connected to each other via a network, the secure computation method comprising:

causing an individual one of the secure computation server apparatuses to compute, by using a shared permutation shared by four of the five secure computation server apparatuses, permuted values of a share for a remaining one of the five secure computation server apparatuses;

causing the individual one of the secure computation server apparatuses to send the permuted values of the share to the remaining secure computation server apparatus;

causing the individual one of the secure computation server apparatuses to compare values with each other, which are received from at least three of the four secure computation server apparatuses and which are supposed to be a same value;

causing the individual one of the secure computation server apparatuses to adopt the values that are same at least two values as an accurate permutation; and

causing the individual one of the secure computation server apparatuses to synthesize, regarding five combinations of four secure computation server apparatuses selected from the five secure computation server apparatuses, mini-shuffles, each of which is constructed by using a shared permutation shared by a corresponding combination of four secure computation server apparatuses and a corresponding permutation adopted.

7. The secure computation method according to claim **6**; wherein the mini-shuffles are each determinably generated by using seeds held by the four secure computation server apparatuses and an identifier held by the five secure computation server apparatuses as input and are each masked by pseudo random numbers whose total is zero regarding the five secure computation server apparatuses.

8. The secure computation method according to claim **6**; wherein the shared permutation is determinably generated by using a seed shared by the four secure computation server apparatuses as input.

9. The secure computation method according to claim **6**; wherein it is determined that the permuted values of the share are each an accurate value by determining that hash values of the received values are same.

10. A non-transient computer readable medium storing a secure computation program, causing at least five secure computation server apparatuses connected to each other via a network to perform a secure computation, the program comprising:

computing, by using a shared permutation shared by four of the five secure computation server apparatuses, permuted values of a share for a remaining one of the five secure computation server apparatuses;

sending the permuted values of the share to the remaining secure computation server apparatus;

comparing values with each other, which are received from at least three of the four secure computation server apparatuses and which are supposed to be a same value;

adopting the values that are same at least two values as an accurate permutation; and

synthesizing, regarding five combinations of four secure computation server apparatuses selected from the five secure computation server apparatuses, mini-shuffles, each of which is constructed by using a shared permutation shared by a corresponding combination of four secure computation server apparatuses and a corresponding permutation adopted.

11. The secure computation server apparatus according to claim **5**; wherein the mini-shuffles are each determinably generated by using seeds held by the four secure computation server apparatuses and an identifier held by the five secure computation server apparatuses as input and are each masked by pseudo random numbers whose total is zero regarding the five secure computation server apparatuses.

12. The secure computation server apparatus according to claim **5**; wherein the shared permutation is determinably generated by using a seed shared by the four secure computation server apparatuses as input.

13. The secure computation server apparatus according to claim **5**; wherein the comparison and verification part determines that the received values are each an accurate value by determining that hash values of the received values are same.

14. The non-transient computer readable medium storing the secure computation program according to claim **10**; wherein the mini-shuffles are each determinably generated by using seeds held by the four secure computation server apparatuses and an identifier held by the five secure computation server apparatuses as input and are each masked by pseudo random numbers whose total is zero regarding the five secure computation server apparatuses.

15. The non-transient computer readable medium storing the secure computation program according to claim **10**; wherein the shared permutation is determinably generated by using a seed shared by the four secure computation server apparatuses as input.

16. The non-transient computer readable medium storing the secure computation program according to claim **10**; wherein it is determined that the permuted values of the share are each an accurate value by determining that hash values of the received values are same.

* * * * *