(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7420147号
(P7420147)

(45)発行日　令和6年1月23日(2024.1.23)

(24)登録日　令和6年1月15日(2024.1.15)

(51)国際特許分類

		FI		
G09C	*1/00 (2006.01)*	G09C	1/00	650Z
H04L	*9/32 (2006.01)*	H04L	9/32	200Z

請求項の数　5　（全26頁）

(21)出願番号　特願2021-561063(P2021-561063)
(86)(22)出願日　令和1年11月28日(2019.11.28)
(86)国際出願番号　PCT/JP2019/046509
(87)国際公開番号　WO2021/106133
(87)国際公開日　令和3年6月3日(2021.6.3)
審査請求日　令和4年5月27日(2022.5.27)

(73)特許権者　000004237
日本電気株式会社
東京都港区芝五丁目７番１号
(74)代理人　100080816
弁理士　加藤 朝道
(74)代理人　100098648
弁理士　内田 潔人
(72)発明者　土田 光
東京都港区芝五丁目７番１号 日本電気株式会社内
審査官　青木 重徳

最終頁に続く

(54)【発明の名称】　シャッフルシステム、シャッフル方法及びプログラム

(57)【特許請求の範囲】
【請求項１】
相互にネットワークで接続された４台の秘密計算ノードを備え、前記４台の秘密計算ノードに秘密分散されたシェアをシャッフルするシャッフルシステムであって、
前記４台の秘密計算ノードのうちの１台の秘密計算ノードを受信ノード、２台の秘密計算ノードを再分散ノード、残る１台の秘密計算ノードを検証ノードとするラウンドを、前記４台の秘密計算ノードで共有された選定順序に従って前記４台の秘密計算ノードの中から前記受信ノード、再分散ノードおよび検証ノードを選定し、
前記再分散ノードが前記受信ノードの知らない置換を用いてそれぞれが保持するシェアを再分散するミニシャッフルを実施し、前記ミニシャッフルの結果を前記受信ノードに送信し、
前記検証ノードが、前記受信ノードの知らない置換を用いて生成した、前記再分散ノードのミニシャッフルの結果を検証するデータを前記受信ノードに送信し、
前記受信ノードは、前記検証ノードから受信したデータを用いた前記ミニシャッフルの正当性を確認した結果、前記ミニシャッフルの正当性を確認できない場合、処理を中止し、正当性を確認できた場合は前記ラウンドを終了して次のラウンドへ進み、
前記ラウンドを、前記４台の秘密計算ノードが少なくとも１回以上受信ノードとなるよう繰り返すことで、
前記シェアのシャッフルを行うシャッフルシステム。
【請求項２】

前記４台の秘密計算ノードは、
３つのシード（ｓｅｅｄ１，ｓｅｅｄ２，ｓｅｅｄ４）と秘密情報ｘのシェア（ｘ１，ｘ２）とを秘密分散して保持する第１の秘密計算ノードと、
３つのシード（ｓｅｅｄ２，ｓｅｅｄ３，ｓｅｅｄ４）と秘密情報ｘのシェア（ｘ２，ｘ３）とを秘密分散して保持する第２の秘密計算ノードと、
３つのシード（ｓｅｅｄ３，ｓｅｅｄ１，ｓｅｅｄ４）と秘密情報ｘのシェア（ｘ３，ｘ１）とを秘密分散して保持する第３の秘密計算ノードと、
３つのシード（ｓｅｅｄ１，ｓｅｅｄ２，ｓｅｅｄ３）と秘密情報ｘのシェア（ｘ１－ｘ２，ｘ２－ｘ３）とを秘密分散して保持する第４の秘密計算ノードと、により構成され、
前記再分散ノード及び前記検証ノードは、前記受信ノードの知らないシードを用いて、２つの乱数を生成し、
前記再分散ノードは、前記ミニシャッフルの結果として、前記シェアに前記乱数の組み合わせを適用した結果を前記受信ノードに送信し、
前記検証ノードは、前記２台の再分散ノードが前記受信ノードに送信した結果の和又は差を計算して、前記受信ノードに送信する、
請求項１のシャッフルシステム。
【請求項３】
前記第４の秘密計算ノードを受信ノードとするラウンドと、第１の秘密計算ノードを受信ノードとするラウンドと、を並列に実行し、
前記第２の秘密計算ノードを受信ノードとするラウンドと、第３の秘密計算ノードを受信ノードとするラウンドと、を並列に実行する、
請求項２のシャッフルシステム。
【請求項４】
相互にネットワークで接続された４台の秘密計算ノードを用いて、前記４台の秘密計算ノードに秘密分散されたシェアをシャッフルするシャッフル方法であって、
前記４台の秘密計算ノードのうちの１台の秘密計算ノードを受信ノード、２台の秘密計算ノードを再分散ノード、残る１台の秘密計算ノードを検証ノードとするラウンドを、前記４台の秘密計算ノードで共有された選定順序に従って前記４台の秘密計算ノードの中から前記受信ノード、再分散ノードおよび検証ノードを選定し、
前記再分散ノードが前記受信ノードの知らない置換を用いてそれぞれが保持するシェアを再分散するミニシャッフルを実施し、前記ミニシャッフルの結果を前記受信ノードに送信し、
前記検証ノードが、前記受信ノードの知らない置換を用いて生成した、前記再分散ノードのミニシャッフルの結果を検証するデータを前記受信ノードに送信し、
前記受信ノードは、前記検証ノードから受信したデータを用いた前記ミニシャッフルの正当性を確認した結果、前記ミニシャッフルの正当性を確認できない場合、処理を中止し、正当性を確認できた場合は前記ラウンドを終了して次のラウンドへ進み、
前記ラウンドを、前記４台の秘密計算ノードが少なくとも１回以上受信ノードとなるよう繰り返すことで、
前記シェアのシャッフルを行うシャッフル方法。
【請求項５】
相互にネットワークで接続された４台の秘密計算ノードに、前記４台の秘密計算ノードに秘密分散されたシェアをシャッフルする処理を行わせるシャッフルプログラムであって、
前記４台の秘密計算ノードのうちの１台の秘密計算ノードを受信ノード、２台の秘密計算ノードを再分散ノード、残る１台の秘密計算ノードを検証ノードとするラウンドを、前記４台の秘密計算ノードで共有された選定順序に従って前記４台の秘密計算ノードの中から前記受信ノード、再分散ノードおよび検証ノードを選定する処理と、
前記再分散ノードが前記受信ノードの知らない置換を用いてそれぞれが保持するシェアを再分散するミニシャッフルを実施し、前記ミニシャッフルの結果を前記受信ノードに送信する処理と、

前記検証ノードが、前記受信ノードの知らない置換を用いて生成した、前記再分散ノードのミニシャッフルの結果を検証するデータを前記受信ノードに送信する処理と、
前記受信ノードが、前記検証ノードから受信したデータを用いた前記ミニシャッフルの正当性を確認した結果、前記ミニシャッフルの正当性を確認できない場合、処理を中止し、正当性を確認できた場合は前記ラウンドを終了して次のラウンドへ進む処理と、
前記ラウンドを、前記４台の秘密計算ノードが少なくとも１回以上受信ノードとなるよう繰り返す処理とを、
含むシャッフルプログラム。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、シャッフルシステム、シャッフル方法及びプログラムに関する。

【背景技術】

【０００２】

特許文献１に、秘密ランダム置換が含まれる秘密計算を高速に行うことができるという秘密計算方法が開示されている。また、秘密計算によるアプリケーションとして、データやクエリを秘匿したデータベース処理が挙げられる。また、この秘密計算上のデータベース演算を実現する際に、重要なサブプロトコルの一つとして，ソートプロトコルが挙げられる。特許文献２には、比較に基づかないソーティングアルゴリズムが開示されている。

【０００３】

上記ソートプロトコルの実現方法はいくつか存在し、代表的なものとして以下の２つが挙げられる。

・ソーティングネットワークによる方式
・シャッフル（ランダム置換）を用いる方式

【０００４】

コストが小さいのはシャッフルを用いる方式であり、特許文献１、２もこの方式を採用している。シャッフル（ランダム置換）を用いる方式の代表的なものとしては、以下の２つが挙げられる。

・３者の（環上の）複製型秘密分散を用いるセミオネスト安全な方式（非特許文献１参照）
・Ｎ者の（体上の）加法型秘密分散を用いるセミオネスト安全／マリシャス安全な方式（非特許文献２参照）

【先行技術文献】

【特許文献】

【０００５】

【文献】国際公開第２０１５／１０７９５２号
【文献】特開２０１２－１５４９９０号公報

【非特許文献】

【０００６】

【文献】五十嵐大， ｅｔ ａｌ．、“超高速秘密計算ソートの設計と実装： 秘密計算がスクリプト言語に並ぶ日”、コンピュータセキュリティシンポジウム２０１７、論文集 ２０１７．２（２０１７）
【文献】Sven Laur, Jan Willemson, and Bingsheng Zhang, “Round-efficient oblivious database manipulation”, International Conference on Information Security, Springer, Berlin, Heidelberg, 2011,［令和元（2019）年11月1日検索］、インターネット〈URL：https://eprint.iacr.org/2011/429.pdf〉

【発明の概要】

【発明が解決しようとする課題】

【０００７】

以下の分析は、本発明者によって与えられたものである。上記した非特許文献２の方式は、コミットメントとゼロ知識証明を用いて、確率的に不正を検知することが可能となっ

ている。しかしながら、非特許文献２の方式をもってしても、決定的に不正を検知可能な方式とはなっていない。

【０００８】

本発明は、決定的に不正を検知可能なシャッフルシステム、シャッフル方法及びプログラムを提供することを目的とする。

【課題を解決するための手段】

【０００９】

第１の視点によれば、４台の秘密計算ノードのうちの１台の秘密計算ノードを受信ノードに選定し、前記４台の秘密計算ノードのうちの残る３台の秘密計算ノードのうち、２台の秘密計算ノードが再分散ノードとして、残る１台の秘密計算ノードが検証ノードとして動作し、前記再分散ノードが前記受信ノードの知らない置換を用いてそれぞれが保持するシェアを再分散するミニシャッフルを実施し、前記ミニシャッフルの結果を前記受信ノードに送信し、前記検証ノードが、前記受信ノードの知らない置換を用いて生成した、前記再分散ノードのミニシャッフルの結果を検証するデータを前記受信ノードに送信することを１ラウンドとする。このラウンドを、前記４台の秘密計算ノードが少なくとも１回以上受信ノードとなるよう繰り返すことで、シェアのシャッフルを行うシャッフルシステムが提供される。

【００１０】

第２の視点によれば、４台の秘密計算ノードのうちの１台の秘密計算ノードを受信ノードに選定し、前記４台の秘密計算ノードのうちの残る３台の秘密計算ノードのうち、２台の秘密計算ノードが再分散ノードとして、残る１台の秘密計算ノードが検証ノードとして動作するシャッフル方法が提供される。このシャッフル方法は、前記再分散ノードが前記受信ノードの知らない置換を用いてそれぞれが保持するシェアを再分散するミニシャッフルを実施し、前記ミニシャッフルの結果を前記受信ノードに送信し、前記検証ノードが、前記受信ノードの知らない置換を用いて生成した、前記再分散ノードのミニシャッフルの結果を検証するデータを前記受信ノードに送信するラウンドを、前記４台の秘密計算ノードが少なくとも１回以上受信ノードとなるよう繰り返すことで、前記シェアのシャッフルを行う。本方法は、４台の秘密計算ノードの構成する秘密計算ノードという、特定の機械に結びつけられている。

【００１１】

第３の視点によれば、上記した秘密計算ノードの機能を実現するための（コンピュータ）プログラムが提供される。このプログラムは、コンピュータ装置に入力装置又は外部から通信インターフェースを介して入力され、記憶装置に記憶されて、プロセッサを所定のステップないし処理に従って駆動させ、必要に応じ中間状態を含めその処理結果を段階毎に表示装置を介して表示することができ、あるいは通信インターフェースを介して、外部と交信することができる。そのためのコンピュータ装置は、一例として、典型的には互いにバスによって接続可能なプロセッサ、記憶装置、入力装置、通信インターフェース、及び必要に応じ表示装置を備える。また、このプログラムは、コンピュータが読み取り可能な（非トランジトリーな）記憶媒体に記録することができる。

【発明の効果】

【００１２】

本発明によれば、決定的に不正を検知可能なシャッフルシステム、シャッフル方法及びプログラムが提供される。

【図面の簡単な説明】

【００１３】

【図１】本発明の一実施形態の構成を示す図である。

【図２】本発明の一実施形態の動作を説明するための図である。

【図３】本発明の一実施形態の動作を説明するための図である。

【図４】本発明の第１の実施形態のシャッフルシステムの構成を示す図である。

【図５】本発明の第１の実施形態の秘密計算サーバの構成を示す図である。

【図６】本発明の第１の実施形態のミニシャッフルの実行順序の一例を示す図である。
【図７】本発明の第１の実施形態のラウンド（ｉ＝１）の動作を説明するための図である。
【図８】本発明の第１の実施形態のラウンド（ｉ＝２）の動作を説明するための図である。
【図９】本発明の第１の実施形態のラウンド（ｉ＝３）の動作を説明するための図である。
【図１０】本発明の第１の実施形態のラウンド（ｉ＝４）の動作を説明するための図である。
【図１１】本発明の第２の実施形態のラウンド（ｉ＝１、２）の動作を説明するための図である。
【図１２】本発明の第２の実施形態のラウンド（ｉ＝３、４）の動作を説明するための図である。
【図１３】本発明の第３の実施形態のラウンド（ｉ＝１）の動作を説明するための図である。
【図１４】本発明の第３の実施形態のラウンド（ｉ＝２）の動作を説明するための図である。
【図１５】本発明の第３の実施形態のラウンド（ｉ＝３）の動作を説明するための図である。
【図１６】本発明の第３の実施形態のラウンド（ｉ＝４）の動作を説明するための図である。
【図１７】本発明の秘密計算サーバを構成するコンピュータの構成を示す図である。
【発明を実施するための形態】
【００１４】
はじめに本発明の一実施形態の概要について図面を参照して説明する。なお、この概要に付記した図面参照符号は、理解を助けるための一例として各要素に便宜上付記したものであり、本発明を図示の態様に限定することを意図するものではない。また、以降の説明で参照する図面等のブロック間の接続線は、双方向及び単方向の双方を含む。一方向矢印については、主たる信号（データ）の流れを模式的に示すものであり、双方向性を排除するものではない。プログラムはコンピュータ装置を介して実行され、コンピュータ装置は、例えば、プロセッサ、記憶装置、入力装置、通信インターフェース、及び必要に応じ表示装置を備える。また、このコンピュータ装置は、通信インターフェースを介して装置内又は外部の機器（コンピュータを含む）と、有線、無線を問わず、通信可能に構成される。また、図中の各ブロックの入出力の接続点には、ポート乃至インターフェースがあるが図示を省略する。また、以下の説明において、「Ａ及び／又はＢ」は、Ａ又はＢ、若しくは、Ａ及びＢという意味で用いる。
【００１５】
本発明は、その一実施形態において、図１に示すように、４台の秘密計算ノード１０－１～１０－４にて構成されたシャッフルシステムにて実現できる。より具体的には、４台の秘密計算ノード１０－１～１０－４のうちの１台の秘密計算ノードを受信ノードに選定し、残る３台の秘密計算ノードのうち、２台の秘密計算ノードが再分散ノードとして、残る１台の秘密計算ノードが検証ノードとして動作する。
【００１６】
例えば、図２に示すように、秘密計算ノード１０－４を受信ノード（Ｒ）に選定したものとする。このとき、秘密計算ノード１０－１、１０－２が再分散ノード（ＲＳ１）、（ＲＳ２）、秘密計算ノード１０－３が検証ノード（Ｖ）として動作することができる。
【００１７】
再分散ノード（ＲＳ１）として動作する秘密計算ノード１０－１、１０－２が、受信ノードである秘密計算ノード１０－４の知らない置換を用いてそれぞれが保持するシェアを再分散するミニシャッフルを実施する。そして、秘密計算ノード１０－１、１０－２は、受信ノードである秘密計算ノード１０－４に対し、前記ミニシャッフルの結果Ｍ１、Ｍ２を送信する。一方、検証ノード（Ｖ）として動作する秘密計算ノード１０－３は、受信ノードである秘密計算ノード１０－４の知らない置換を用いて、秘密計算ノード１０－１、

１０－２のミニシャッフルの結果Ｍ１、Ｍ２を検証するデータＶを計算する。そして、秘密計算ノード１０－３は、秘密計算ノード１０－４に対し、秘密計算ノード１０－１、１０－２のミニシャッフルの結果Ｍ１、Ｍ２を検証するデータＶを送信する。そして、秘密計算ノード１０－４は、ミニシャッフルの結果Ｍ１、Ｍ２を検証するデータＶを用いて、秘密計算ノード１０－１、１０－２によるミニシャッフルが正しく行われているか否かを検査する。

【００１８】

以上の各手順を１ラウンドとする。図３は、秘密計算ノード１０－１を受信ノード（Ｒ）に選定し、秘密計算ノード１０－２、１０－３が再分散ノード（ＲＳ１）、（ＲＳ２）、秘密計算ノード１０－４が検証ノード（Ｖ）として動作するラウンドを示している。この場合も同様に、秘密計算ノード１０－２、１０－３がミニシャッフルを行い、秘密計算ノード１０－１に対し、ミニシャッフルの結果Ｍ１、Ｍ２を送信する。秘密計算ノード１０－４は、秘密計算ノード１０－１に対し、ミニシャッフルの結果Ｍ１、Ｍ２を検証するデータＶを送信する。そして、秘密計算ノード１０－１が、前記データＶを用いて、秘密計算ノード１０－２、１０－３によるミニシャッフルが正しく行われているか否かを検査する。

【００１９】

以上のようにして、各秘密計算ノードが少なくとも１回以上受信ノードとなるよう繰り返すことで、シェアのシャッフルを行うことができる。さらに、前述のように各ラウンドで、受信ノード（Ｒ）が、検証ノード（Ｖ）にて作成されたデータを用いて、ミニシャッフルが正しく行われているか否かを検査するため、決定的に不正を検知することが可能となる。

【００２０】

［第１の実施形態］

続いて、本発明の第１の実施形態について図面を参照して詳細に説明する。図４は、本発明の第１の実施形態のシャッフルシステムの構成を示す図である。図４を参照すると、上述した秘密計算ノードとして機能する４台の秘密計算サーバＰ$_1$～Ｐ$_4$が相互に接続された構成が示されている。以下、秘密計算サーバを特に指定しない場合、秘密計算サーバＰ$_i$とも記す。

【００２１】

ここで、以下の説明で用いる表記について定義する。なお、ｆｉｎｉｔｅ　ｒｉｎｇは、有限環、ｐｓｅｕｄｏｒａｎｄｏｍ　ｆｕｎｃｔｉｏｎは疑似乱数関数を表す。また、Ｓ$_m$は、ｍ個の要素に対する置換の集合を表し、πは、Ｓ$_m$に属する任意の置換である。［π］$_i$は、Ｓ$_m$に属するｍ個の要素の置換セットのうち、Ｐ$_i$、Ｐ$_{i+1}$、Ｐ$_{i+2}$のみが知る置換を表す。

【００２２】

【数１】

The i-th party: P_i $(i = 1,2,3,4)$
Security parameter: κ
The i-th seed: $seed_i \in \{0,1\}^\kappa$
A finite ring: $\mathbb{R}$ $s.t.$ $|\mathbb{R}|$ is k-bits and its order is R.
A pseudorandom function: $H: \{0,1\}^\kappa \times \{0,1\}^\kappa \rightarrow \{0,1\}^k$
The vector of shares: $[\vec{x}]^m = ([x_1], \ldots, [x_m])$
All permutations of a set with m elements: S_m
A permutation of a set with m elements: $\pi \in S_m$
Only P_i, P_{i+1} and P_{i+2} know a permutation π: $[\pi]_i$

以降、任意の要素xのベクトルについては、上記xに上付きの矢線の表記のほか、［ｖｅｃ｛ｘ｝］とも記す。例えば、上記［数１］中のシェアｘのｍ次元のベクトルは、［ｖｅｃ｛ｘ｝］^{m}とも表記する。

【００２３】

図５は、本発明の第１の実施形態の秘密計算サーバの構成を示す図である。図５を参照すると、置換生成部１０１と、置換適用部１０２と、算術演算部１０３と、不正検知部１０４と、乱数計算部１０５と、ハッシュ値計算部１０６と、シード記憶部１０７と、シェア値記憶部１０８とを備えた構成が示されている。

【００２４】

置換生成部１０１は、他の秘密計算サーバP_iと連携して、P_i、P_{i+1}、P_{i+2}のみが知る置換$[\pi]_i$を生成する。例えば、秘密計算サーバP_1は$[\pi]_1$、$[\pi]_3$、$[\pi]_4$を保持するが、$[\pi]_2$に関する情報は保持しないことになる。

【００２５】

置換適用部１０２は、上記シェアｘのｍ次元ベクトル$[vec\{x\}]^m$と、置換$[\pi]_i$を入力として、$[vec\{x\}]^m$を置換した$[vec\{y\}]^m$を出力する。置換適用部１０２の具体的な処理の例は、後に、本実施形態の動作とともに説明する。

【００２６】

算術演算部１０３は、ｘのシェアのｍ次元ベクトル$[vec\{x\}]^m$と、乱数計算部１０５で計算された乱数との計算等を行う。算術演算部１０３の具体的な処理の例は、後に、本実施形態の動作とともに説明する。

【００２７】

不正検知部１０４は、受信ノードとして動作する際に、検証ノードから送られた検証用データを用いて、他の秘密計算サーバで置換が正しく行われているか否かを検知する。他の秘密計算サーバで置換が正しく行われていないと判定した場合、不正検知部１０４は、処理の中止を決定する。

【００２８】

乱数計算部１０５は、シード記憶部１０７に保持されているシードを用いて、２つの乱数を生成し、ハッシュ値計算部１０６に送る。

【００２９】

ハッシュ値計算部１０６は、乱数計算部１０５にて計算された乱数のハッシュ値を計算する。本実施形態では、このハッシュ値を乱数として使用する。

【００３０】

シード記憶部１０７は、上記した乱数を生成するためのシードを記憶する。本実施形態では、事前に、秘密計算サーバP_1～P_4に、以下のようにシードが配布されているものとする。

P_1：（$seed_1$，$seed_2$，$seed_4$）
P_2：（$seed_2$，$seed_3$，$seed_4$）
P_3：（$seed_3$，$seed_1$，$seed_4$）
P_4：（$seed_1$，$seed_2$，$seed_3$）

【００３１】

シェア値記憶部１０８は、置換の対象となるシェアのｍ次元ベクトルを記憶する。以下の説明では、各秘密計算サーバP_1～P_4は、以下に示すように、２－ｏｕｔ－ｏｆ－４の秘密分散方式でｘのシェアを分散保持するものとして説明する。ここでｘは、有限環Ｒの元であり、x_1，x_2，x_3は、$x_1+x_2+x_3$がｘ（ｍｏｄ　Ｒ）を満たすようにランダムに生成される。以下、ｘのシェアを［ｘ］と表記し、秘密計算サーバP_iが保持するシェアを［ｘ］ｉと表記する。

P_1：$[x]_1=(x_1, x_2)$
P_2：$[x]_2=(x_2, x_3)$
P_3：$[x]_3=(x_3, x_1)$
P_4：$[x]_4=(x_1-x_2, x_2-x_3)$

【００３２】
続いて、本実施形態の動作について図面を参照して詳細に説明する。本実施形態では、秘密計算サーバP_1～P_4のうちの１台が受信ノード、２台が再分散ノード、１台が検証ノードとして動作してミニシャッフルを行うラウンドを少なくとも４回繰り返すことで、ｘのシェアのｍ次元ベクトル［ｖｅｃ｛ｘ｝］^{m}のシャッフルを行う。すべてのラウンドで、不正検知部１０４が不正を検出できなかった場合、シャッフル成功となり、いずれかのラウンドで、不正検知部１０４が不正を検出した場合、処理を中止する。
【００３３】
図６は、本発明の第１の実施形態のミニシャッフルの実行順序の一例を示す図である。図６を参照すると、まず、ｉ＝１として、P_{i+3}であるP_4が受信ノードとなる。そして、P_i、P_{i+1}、P_{i+2}にあたるP_1～P_3が再分散ノード及び検証ノードとしてミニシャッフルを行う。以下、同様に、ｉをインクリメントしていき、ｉ＝４のラウンドを実行することでシャッフルが完了する。
【００３４】
続いて、各ラウンドにおいて行われる処理の詳細を順番に説明する。
［ラウンド１　ｉ＝１］
図７は、ラウンド（ｉ＝１）の動作を説明するための図である。ラウンド１においては、秘密計算サーバP_4が受信ノードとなる。また、秘密計算サーバP_1，P_2が再分散ノード、秘密計算サーバP_3が検証ノードとなる。ラウンド１におけるミニシャッフルを次式［数２］のように表すものとする。
【００３５】
【数２】

$$[\vec{y}]^m \leftarrow MiniShuffle(1, [\vec{x}]^m, [\pi_1]_1)$$

右辺の［ｖｅｃ｛ｘ｝］^{m}は、シェアｘを秘密分散したｍ次元のベクトルであり、［x_1］，・・・，［$x_{\underline{m}}$］のように表される。実際には、前述のように、x_jは、$x_j = x_{j,1} + x_{j,2} + x_{j,3}$（ｊ＝１，・・・，ｍ）を満たす形で、秘密計算サーバP_iに秘密分散される。右辺の［π］$_1$は、秘密計算サーバP_1、P_2、P_3のみが知る置換［π］$_i$である。
【００３６】
以上の右辺の内容を入力とする出力［ｖｅｃ｛ｙ｝］^{m}は、置換［π］$_1$を用いたミニシャッフルを１回行った結果であり、［$y_{\pi 1(1)}$］，・・・，［$y_{\pi 1(m)}$］のように表される。この出力中の$y_{\pi 1(j)}$も、前述のように、$y_{\pi 1(j)} = y_{\pi 1(j),1} + y_{\pi 1(j),2} + y_{\pi 1(j),3}$（ｊ＝１，・・・，ｍ）を満たす形で、秘密計算サーバP_iに分散保持される。
【００３７】
次式［数３］に上記したミニシャッフルの手順を示す。
【００３８】
【数３】

1. P_1, P_2 and P_3 generate the random values $r_{j,1}$ and $r_{j,2}$.
   $r_{j,1} = H(vid_1||1||j,\ seed_4)$
   $r_{j,2} = H(vid_1||2||j,\ seed_4)$
2. We set $y_{\pi_1(j)} = y_{\pi_1(j),1} + y_{\pi_1(j),2} + y_{\pi_1(j),3}$.
   2-1. P_1 computes $y_{\pi_1(j),1}$ and $y_{\pi_1(j),2}$ as follows.
   $y_{\pi_1(j),1} = x_{\pi_1(j),1} - r_{j,1}$
   $y_{\pi_1(j),2} = x_{\pi_1(j),2} + r_{j,1} + r_{j,2}$
   2-2. P_2 computes $y_{\pi_1(j),2}$ and $y_{\pi_1(j),3}$ as follows.
   $y_{\pi_1(j),2} = x_{\pi_1(j),2} + r_{j,1} + r_{j,2}$
   $y_{\pi_1(j),3} = x_{\pi_1(j),3} - r_{j,2}$
   2-3. P_3 computes $y_{\pi_1(j),3}$ and $y_{\pi_1(j),1}$ as follows.
   $y_{\pi_1(j),3} = x_{\pi_1(j),3} - r_{j,2}$
   $y_{\pi_1(j),1} = x_{\pi_1(j),1} - r_{j,1}$
3. P_1 sends $\overrightarrow{m_{1,1}} = \overrightarrow{y_1} - \overrightarrow{y_2}$ to P_4.
   P_2 sends $\overrightarrow{m_{2,2}} = \overrightarrow{y_2} - \overrightarrow{y_3}$ to P_4.
   P_3 sends $\overrightarrow{m_3} = \overrightarrow{y_1} - \overrightarrow{y_3}$ to P_4.
   $\overrightarrow{y_i} = (y_{\pi_1(1),i}, \cdots, y_{\pi_1(m),i})$
4. P_4 checks whether $\overrightarrow{m_3} = \overrightarrow{m_{1,1}} + \overrightarrow{m_{2,2}}$ or not.
   If it holds, then P_4 outputs *continue*.
   If it does not hold, then P_4 outputs *abort*.

【００３９】
（ステップ１－１）まず、秘密計算サーバＰ１、Ｐ２、Ｐ３は、受信ノードである秘密計算サーバＰ４が知らないシードｓｅｅｄ４を用いて、２つの乱数ｒｊ，１、ｒｊ，２を生成する。
【００４０】
（ステップ１－２－１～１－２－３）次に、秘密計算サーバＰ１、Ｐ２、Ｐ３は、互いに連携して、乱数ｒｊ，１、ｒｊ，２を用いて、ｙπ１（ｊ）＝ｙπ１（ｊ），１＋ｙπ１（ｊ），２＋ｙπ１（ｊ），３を満たすようなｙπ１（ｊ），１，ｙπ１（ｊ），２，ｙπ１（ｊ），３をそれぞれ計算する。具体的には、ｙπ１（ｊ），１，ｙπ１（ｊ），２，ｙπ１（ｊ），３は、以下のように計算される。

ｙπ１（ｊ），１＝ｘπ１（ｊ），１－ｒｊ，１
ｙπ１（ｊ），２＝ｘπ１（ｊ），２＋ｒｊ，１＋ｒｊ，２
ｙπ１（ｊ），３＝ｘπ１（ｊ），３－ｒｊ，２

【００４１】
（ステップ１－３）次に、秘密計算サーバＰ１、Ｐ２、Ｐ３は、受信ノードである秘密計算サーバＰ４に対し、計算結果を送信する。具体的には、秘密計算サーバＰ１は、秘密計算サーバＰ４に対し、ｖｅｃ｛ｙ１｝－ｖｅｃ｛ｙ２｝をｖｅｃ｛ｍ１，１｝として送信する。秘密計算サーバＰ２は、秘密計算サーバＰ４に対し、ｖｅｃ｛ｙ２｝－ｖｅｃ｛ｙ３｝をｖｅｃ｛ｍ２，２｝として送信する。さらに、秘密計算サーバＰ３は、秘密計算サーバＰ４に対し、検証データとして、ｖｅｃ｛ｙ１｝－ｖｅｃ｛ｙ３｝をｖｅｃ｛ｍ３｝として送信する。なお、［数３］に表したとおり、ｖｅｃ｛ｙｉ｝は、（［ｙπ１（１），ｉ］，・・・，［ｙπ１（ｍ），ｉ］）のようなｍ次元のベクトルである。
【００４２】
（ステップ１－４）次に、秘密計算サーバＰ４は、ｖｅｃ｛ｍ３｝＝ｖｅｃ｛ｍ１，１｝＋ｖｅｃ｛ｍ２，２｝が成り立つか否かにより、不正検知を行う。ｖｅｃ｛ｍ３｝＝ｖｅｃ｛ｍ１，１｝＋ｖｅｃ｛ｍ２，２｝が成り立つ場合、正しくミニシャッフルが行われたと判定し、秘密計算サーバＰ４は、次のラウンドに進む。一方、ｖｅｃ｛ｍ３｝＝ｖｅｃ｛ｍ１，１｝＋ｖｅｃ｛ｍ２，２｝が成り立たない場合、秘密計算サーバＰ４は、不正なシャッフルが行われたと判定し、以降の処理を中断する。
【００４３】

以上のミニシャッフルの結果、秘密計算サーバＰ１～Ｐ４は、次式［数４］のように、ミニシャッフル後のｍ次元ベクトルを保持することになる。

【００４４】

【数４】

$$
\begin{aligned}
P_1&:(\overrightarrow{y_1},\overrightarrow{y_2})=\left(\left(y_{\pi_1(1),1},y_{\pi_1(1),2}\right),\cdots,\left(y_{\pi_1(m),1},y_{\pi_1(m),2}\right)\right)=\left(\left[y_{\pi_1(1)}\right]_1,\cdots,\left[y_{\pi_1(m)}\right]_1\right)\\
P_2&:(\overrightarrow{y_2},\overrightarrow{y_3})=\left(\left(y_{\pi_1(1),2},y_{\pi_1(1),3}\right),\cdots,\left(y_{\pi_1(m),2},y_{\pi_1(m),3}\right)\right)=\left(\left[y_{\pi_1(1)}\right]_2,\cdots,\left[y_{\pi_1(m)}\right]_2\right)\\
P_3&:(\overrightarrow{y_3},\overrightarrow{y_1})=\left(\left(y_{\pi_1(1),3},y_{\pi_1(1),1}\right),\cdots,\left(y_{\pi_1(m),3},y_{\pi_1(m),1}\right)\right)=\left(\left[y_{\pi_1(1)}\right]_3,\cdots,\left[y_{\pi_1(m)}\right]_3\right)\\
P_4&:\left(\overrightarrow{m_{1,1}},\overrightarrow{m_{2,2}}\right)\\
&\quad=(\overrightarrow{y_1}-\overrightarrow{y_2},\overrightarrow{y_2}-\overrightarrow{y_3})\\
&\quad=\left(\left(y_{\pi_1(1),1}-y_{\pi_1(1),2},y_{\pi_1(1),2}-y_{\pi_1(1),3}\right),\cdots,\left(y_{\pi_1(m),1}-y_{\pi_1(m),2},y_{\pi_1(m),2}-y_{\pi_1(m),3}\right)\right)\\
&\quad=\left(\left[y_{\pi_1(1)}\right]_4,\cdots,\left[y_{\pi_1(m)}\right]_4\right)
\end{aligned}
$$

【００４５】

［ラウンド２　ｉ＝２］

図８は、ラウンド（ｉ＝２）の動作を説明するための図である。ラウンド２においては、秘密計算サーバＰ１が受信ノードとなる。また、秘密計算サーバＰ２，Ｐ３が再分散ノード、秘密計算サーバＰ４が検証ノードとなる。ラウンド２におけるミニシャッフルを次式［数５］のように表すものとする。

【００４６】

【数５】

$$[\vec{y}]^m \leftarrow MiniShuffle(2,[\vec{x}]^m,[\pi_2]_2)$$

右辺の［π］２は、秘密計算サーバＰ２、Ｐ３、Ｐ４のみが知る置換［π］ｉである。

【００４７】

以上の右辺の内容を入力とする出力［ｖｅｃ｛ｙ｝］ｍは、置換［π］２を用いたミニシャッフルを１回行った結果であり、［ｙπ２（１）］，・・・，［ｙπ２（ｍ）］のように表される。この出力中のｙπ２（ｊ）も、前述のように、ｙπ２（ｊ）＝ｙπ２（ｊ），１＋ｙπ２（ｊ），２＋ｙπ２（ｊ），３（ｊ＝１，・・・，ｍ）を満たす形で、秘密計算サーバＰｉに分散保持される。

【００４８】

次式［数６］に上記したミニシャッフルの手順を示す。

【００４９】

【数６】

1. P_2, P_3 and P_4 generate the random values $r_{j,1}$ and $r_{j,2}$.
   $r_{j,1} = H(vid_1||1||j,\ seed_3)$
   $r_{j,2} = H(vid_1||2||j,\ seed_3)$
2. We set $y_{\pi_2(j)} = y_{\pi_2(j),1} + y_{\pi_2(j),2} + y_{\pi_2(j),3}$.
   2-1. P_2 computes $y_{\pi_2(j),2}$ and $y_{\pi_2(j),3}$ as follows.
   $y_{\pi_2(j),2} = x_{\pi_2(j),2} + r_{j,1} + r_{j,2}$
   $y_{\pi_2(j),3} = x_{\pi_2(j),3} - r_{j,2}$
   2-2. P_3 computes $y_{\pi_2(j),1}$ and $y_{\pi_2(j),3}$ as follows.
   $y_{\pi_2(j),1} = x_{\pi_2(j),1} - r_{j,1}$
   $y_{\pi_2(j),3} = x_{\pi_2(j),3} - r_{j,2}$
   2-3. P_4 computes $y_{\pi_2(j),1} - y_{\pi_2(j),2}$ and $y_{\pi_2(j),2} - y_{\pi_2(j),3}$ as follows.
   $y_{\pi_2(j),1} - y_{\pi_2(j),2} = \left(x_{\pi_2(j),1} - x_{\pi_2(j),2}\right) - 2r_{j,1} - r_{j,2}$
   $y_{\pi_2(j),2} - y_{\pi_2(j),3} = \left(x_{\pi_2(j),2} - x_{\pi_2(j),3}\right) + r_{j,1} + 2r_{j,2}$
3. P_2 sends $\overrightarrow{m_{2,2}} = \overrightarrow{y_2}$ to P_1.
   P_3 sends $\overrightarrow{m_{1,3}} = \overrightarrow{y_1}$ to P_1.
   P_4 sends $\overrightarrow{m_4} = \overrightarrow{y_1} - \overrightarrow{y_2}$ to P_1.
   $\overrightarrow{y_i} = (y_{\pi_2(1),i}, \ldots, y_{\pi_2(m),i})$
4. P_1 checks whether $\overrightarrow{m_4} = \overrightarrow{m_{1,3}} - \overrightarrow{m_{2,2}}$ or not.
   If it holds, then P_1 outputs continue.
   If it does not hold, then P_1 outputs abort.

【００５０】
（ステップ２－１）まず、秘密計算サーバＰ２、Ｐ３、Ｐ４は、受信ノードである秘密計算サーバＰ１が知らないシードｓｅｅｄ３を用いて、２つの乱数ｒｊ，１、ｒｊ，２を生成する。
【００５１】
（ステップ２－２－１～２－２－３）次に、秘密計算サーバＰ２、Ｐ３、Ｐ４は、互いに連携して、乱数ｒｊ，１、ｒｊ，２を用いて、ｙπ２（ｊ）＝ｙπ２（ｊ），１＋ｙπ２（ｊ），２＋ｙπ２（ｊ），３（ｍｏｄ　Ｒ）を満たすようなｙπ２（ｊ），１，ｙπ２（ｊ），２，ｙπ２（ｊ），３を計算する。具体的には、ｙπ２（ｊ），１，ｙπ２（ｊ），２，ｙπ２（ｊ），３は、以下のように計算される。

ｙπ２（ｊ），１＝ｘπ２（ｊ），１－ｒｊ，１
ｙπ２（ｊ），２＝ｘπ２（ｊ），２＋ｒｊ，１＋ｒｊ，２
ｙπ２（ｊ），３＝ｘπ２（ｊ），３－ｒｊ，２

【００５２】
さらに、秘密計算サーバＰ４は、［数６］に示すように、ｙπ２（ｊ），１－ｙπ２（ｊ），２、ｙπ２（ｊ），２－ｙπ２（ｊ），３を計算する。
【００５３】
（ステップ２－３）次に、秘密計算サーバＰ２、Ｐ３、Ｐ４は、受信ノードである秘密計算サーバＰ１に対し、計算結果を送信する。具体的には、秘密計算サーバＰ２は、秘密計算サーバＰ１に対し、ｖｅｃ｛ｙ２｝をｖｅｃ｛ｍ２，２｝として送信する。秘密計算サーバＰ３は、秘密計算サーバＰ１に対し、ｖｅｃ｛ｙ１｝をｖｅｃ｛ｍ１，３｝として送信する。さらに、秘密計算サーバＰ４は、秘密計算サーバＰ１に対し、検証データとして、ｖｅｃ｛ｙ１｝－ｖｅｃ｛ｙ２｝をｖｅｃ｛ｍ４｝として送信する。なお、［数６］に表したとおり、ｖｅｃ｛ｙｉ｝は、（［ｙπ２（１），ｉ］，・・・，［ｙπ２（ｍ），ｉ］）のようなｍ次元のベクトルである。
【００５４】
（ステップ２－４）次に、秘密計算サーバＰ１は、ｖｅｃ｛ｍ４｝＝ｖｅｃ｛ｍ１，３｝＋ｖｅｃ｛ｍ２，２｝が成り立つか否かにより、不正検知を行う。ｖｅｃ｛ｍ４｝＝ｖｅｃ｛ｍ１，３｝＋ｖｅｃ｛ｍ２，２｝が成り立つ場合、正しくミニシャッフルが行われたと判定し、秘密計算サーバＰ１は、次のラウンドに進む。一方、ｖｅｃ｛ｍ４｝＝ｖｅｃ｛ｍ１，

３｝＋ｖｅｃ｛ｍ２，２｝が成り立たない場合、秘密計算サーバＰ１は、不正なシャッフルが行われたと判定し、以降の処理を中断する。

【００５５】

以上のミニシャッフルの結果、秘密計算サーバＰ１～Ｐ４は、次式［数７］のように、ミニシャッフル後のｍ次元ベクトルを保持することになる。

【００５６】

【数７】

$$
\begin{aligned}
P_1&: (\overrightarrow{m_{1,3}}, \overrightarrow{m_{2,2}}) \\
&= (\overrightarrow{y_1}, \overrightarrow{y_2}) \\
&= \left(\left(y_{\pi_2(1),1}, y_{\pi_2(1),2}\right), \cdots, \left(y_{\pi_2(m),1}, y_{\pi_2(m),2}\right)\right) = \left(\left[y_{\pi_2(1)}\right]_1, \cdots, \left[y_{\pi_2(m)}\right]_1\right) \\
P_2&: (\overrightarrow{y_2}, \overrightarrow{y_3}) = \left(\left(y_{\pi_2(1),2}, y_{\pi_2(1),3}\right), \cdots, \left(y_{\pi_2(m),2}, y_{\pi_2(m),3}\right)\right) = \left(\left[y_{\pi_2(1)}\right]_2, \cdots, \left[y_{\pi_2(m)}\right]_2\right) \\
P_3&: (\overrightarrow{y_3}, \overrightarrow{y_1}) = \left(\left(y_{\pi_2(1),3}, y_{\pi_2(1),1}\right), \cdots, \left(y_{\pi_2(m),3}, y_{\pi_2(m),1}\right)\right) = \left(\left[y_{\pi_2(1)}\right]_3, \cdots, \left[y_{\pi_2(m)}\right]_3\right) \\
P_4&: (\overrightarrow{y_1} - \overrightarrow{y_2}, \overrightarrow{y_2} - \overrightarrow{y_3}) \\
&= \left(\left(y_{\pi_2(1),1} - y_{\pi_2(1),2}, y_{\pi_2(1),2} - y_{\pi_2(1),3}\right), \cdots, \left(y_{\pi_2(m),1} - y_{\pi_2(m),2}, y_{\pi_2(m),2} - y_{\pi_2(m),3}\right)\right) \\
&= \left(\left[y_{\pi_2(1)}\right]_4, \cdots, \left[y_{\pi_2(m)}\right]_4\right)
\end{aligned}
$$

【００５７】

［ラウンド３　ｉ＝３］

図９は、ラウンド（ｉ＝３）の動作を説明するための図である。ラウンド３においては、秘密計算サーバＰ２が受信ノードとなる。また、秘密計算サーバＰ３，Ｐ１が再分散ノード、秘密計算サーバＰ４が検証ノードとなる。ラウンド３におけるミニシャッフルを次式［数８］のように表すものとする。

【００５８】

【数８】

$$[\vec{y}]^m \leftarrow MiniShuffle(3, [\vec{x}]^m, [\pi_3]_3)$$

右辺の［π］３は、秘密計算サーバＰ１、Ｐ３、Ｐ４のみが知る置換［π］ｉである。

【００５９】

以上の右辺の内容を入力とする出力［ｖｅｃ｛ｙ｝］ｍは、置換［π］３を用いたミニシャッフルを１回行った結果であり、［ｙπ３（１）］，・・・，［ｙπ３（ｍ）］のように表される。この出力中のｙπ３（ｊ）も、前述のように、ｙπ３（ｊ）＝ｙπ３（ｊ），１＋ｙπ３（ｊ），２＋ｙπ３（ｊ），３（ｊ＝１，・・・，ｍ）を満たす形で、秘密計算サーバＰｉに分散保持される。

【００６０】

次式［数９］に上記したミニシャッフルの手順を示す。

【００６１】

【数９】

1. P_3, P_4 and P_1 generate the random values $r_{j,1}$ and $r_{j,2}$.
$r_{j,1} = H(vid_1||1||j,\ seed_1)$
$r_{j,2} = H(vid_1||2||j,\ seed_1)$
2. We set $y_{\pi_3(j)} = y_{\pi_3(j),1} + y_{\pi_3(j),2} + y_{\pi_3(j),3}$.
2-1. P_3 computes $y_{\pi_3(j),1}$ and $y_{\pi_3(j),3}$ as follows.
$y_{\pi_3(j),1} = x_{\pi_3(j),1} - r_{j,1}$
$y_{\pi_3(j),3} = x_{\pi_3(j),3} - r_{j,2}$
2-2. P_1 computes $y_{\pi_3(j),1}$ and $y_{\pi_3(j),2}$ as follows.
$y_{\pi_3(j),1} = x_{\pi_3(j),1} - r_{j,1}$
$y_{\pi_3(j),2} = x_{\pi_3(j),2} + r_{j,1} + r_{j,2}$
2-3. P_4 computes $y_{\pi_3(j),1} - y_{\pi_3(j),2}$ and $y_{\pi_3(j),2} - y_{\pi_3(j),3}$ as follows:
$y_{\pi_3(j),1} - y_{\pi_3(j),2} = \left(x_{\pi_3(j),1} - x_{\pi_3(j),2}\right) - 2r_{j,1} - r_{j,2}$
$y_{\pi_3(j),2} - y_{\pi_3(j),3} = \left(x_{\pi_3(j),2} - x_{\pi_3(j),3}\right) + r_{j,1} + 2r_{j,2}$
3. P_1 sends $\overrightarrow{m_{2,1}} = \overrightarrow{y_2}$ to P_2.
P_3 sends $\overrightarrow{m_{3,3}} = \overrightarrow{y_3}$ to P_2.
P_4 sends $\overrightarrow{m_4} = \overrightarrow{y_2} - \overrightarrow{y_3}$ to P_2.
$\overrightarrow{y_i} = (y_{\pi_3(1),i}, \cdots, y_{\pi_3(m),i})$
4. P_2 checks whether $\overrightarrow{m_4} = \overrightarrow{m_{2,1}} - \overrightarrow{m_{3,3}}$ or not.
If it holds, then P_2 outputs continue.
If it does not hold, then P_2 outputs abort.

【００６２】
（ステップ３－１）まず、秘密計算サーバＰ３、Ｐ４、Ｐ１は、受信ノードである秘密計算サーバＰ２が知らないシードｓｅｅｄ１を用いて、２つの乱数ｒｊ，１、ｒｊ，２を生成する。
【００６３】
（ステップ３－２－１～３－２－３）次に、秘密計算サーバＰ３、Ｐ４、Ｐ１は、互いに連携して、乱数ｒｊ，１、ｒｊ，２を用いて、ｙπ３（ｊ）＝ｙπ３（ｊ），１＋ｙπ３（ｊ），２＋ｙπ３（ｊ），３（ｍｏｄ　Ｒ）を満たすようなｙπ３（ｊ），１，ｙπ３（ｊ），２，ｙπ３（ｊ），３を計算する。具体的には、ｙπ３（ｊ），１，ｙπ３（ｊ），２，ｙπ３（ｊ），３は、以下のように計算される。

ｙπ３（ｊ），１＝ｘπ３（ｊ），１－ｒｊ，１
ｙπ３（ｊ），２＝ｘπ３（ｊ），２＋ｒｊ，１＋ｒｊ，２
ｙπ３（ｊ），３＝ｘπ３（ｊ），３－ｒｊ，２

【００６４】
さらに、秘密計算サーバＰ４は、［数９］に示すように、ｙπ３（ｊ），１－ｙπ３（ｊ），２、ｙπ３（ｊ），２－ｙπ３（ｊ），３を計算する。
【００６５】
（ステップ３－３）次に、秘密計算サーバＰ３、Ｐ４、Ｐ１は、受信ノードである秘密計算サーバＰ２に対し、計算結果を送信する。具体的には、秘密計算サーバＰ３は、秘密計算サーバＰ２に対し、ｖｅｃ｛ｙ３｝をｖｅｃ｛ｍ３，３｝として送信する。秘密計算サーバＰ１は、秘密計算サーバＰ２に対し、ｖｅｃ｛ｙ２｝をｖｅｃ｛ｍ２，１｝として送信する。さらに、秘密計算サーバＰ４は、検証データとして、秘密計算サーバＰ２に対し、ｖｅｃ｛ｙ２｝－ｖｅｃ｛ｙ３｝をｖｅｃ｛ｍ４｝として送信する。なお、［数９］に表したとおり、ｖｅｃ｛ｙｉ｝は、（［ｙπ３（１），ｉ］，・・・，［ｙπ３（ｍ），ｉ］）のようなｍ次元のベクトルである。
【００６６】
（ステップ３－４）次に、秘密計算サーバＰ２は、ｖｅｃ｛ｍ４｝＝ｖｅｃ｛ｍ２，１｝－ｖｅｃ｛ｍ３，３｝が成り立つか否かにより、不正検知を行う。ｖｅｃ｛ｍ４｝＝ｖｅｃ｛ｍ２，１｝－ｖｅｃ｛ｍ３，３｝が成り立つ場合、正しくミニシャッフルが行われたと判定し、秘密計算サーバＰ２は、次のラウンドに進む。一方、ｖｅｃ｛ｍ４｝＝ｖｅｃ｛ｍ２，

１｝－ｖｅｃ｛ｍ３，３｝が成り立たない場合、秘密計算サーバＰ２は、不正なシャッフルが行われたと判定し、以降の処理を中断する。

【００６７】

以上のミニシャッフルの結果、秘密計算サーバＰ１～Ｐ４は、次式［数１０］のように、ミニシャッフル後のｍ次元ベクトルを保持することになる。

【００６８】

【数１０】

$$
\begin{aligned}
&P_1: (\overrightarrow{y_1}, \overrightarrow{y_2}) = \left(\left(y_{\pi_3(1),1}, y_{\pi_3(1),2} \right), \cdots, \left(y_{\pi_3(m),1}, y_{\pi_3(m),2} \right) \right) = \left(\left[y_{\pi_3(1)} \right]_1, \cdots, \left[y_{\pi_3(m)} \right]_1 \right) \\
&P_2: (\overrightarrow{m_{2,1}}, \overrightarrow{m_{3,3}}) \\
&\quad = (\overrightarrow{y_2}, \overrightarrow{y_3}) \\
&\quad = \left(\left(y_{\pi_3(1),2}, y_{\pi_3(1),3} \right), \cdots, \left(y_{\pi_3(m),2}, y_{\pi_3(m),3} \right) \right) = \left(\left[y_{\pi_3(1)} \right]_2, \cdots, \left[y_{\pi_3(m)} \right]_2 \right) \\
&P_3: (\overrightarrow{y_3}, \overrightarrow{y_1}) = \left(\left(y_{\pi_3(1),3}, y_{\pi_3(1),1} \right), \cdots, \left(y_{\pi_3(m),3}, y_{\pi_3(m),1} \right) \right) = \left(\left[y_{\pi_3(1)} \right]_3, \cdots, \left[y_{\pi_3(m)} \right]_3 \right) \\
&P_4: (\overrightarrow{y_1} - \overrightarrow{y_2}, \overrightarrow{y_2} - \overrightarrow{y_3}) \\
&\quad = \left(\left(y_{\pi_3(1),1} - y_{\pi_3(1),2}, y_{\pi_3(1),2} - y_{\pi_3(1),3} \right), \cdots, \left(y_{\pi_3(m),1} - y_{\pi_3(m),2}, y_{\pi_3(m),2} - y_{\pi_3(m),3} \right) \right) \\
&\quad = \left(\left[y_{\pi_3(1)} \right]_4, \cdots, \left[y_{\pi_3(m)} \right]_4 \right)
\end{aligned}
$$

【００６９】

［ラウンド４　ｉ＝４］

図１０は、ラウンド（ｉ＝４）の動作を説明するための図である。ラウンド４においては、秘密計算サーバＰ３が受信ノードとなる。また、秘密計算サーバＰ１，Ｐ２が再分散ノード、秘密計算サーバＰ４が検証ノードとなる。ラウンド４におけるミニシャッフルを次式［数１１］のように表すものとする。

【００７０】

【数１１】

$$[\vec{y}]^m \leftarrow MiniShuffle(4, [\vec{x}]^m, [\pi_4]_4)$$

右辺の［π］４は、秘密計算サーバＰ１、Ｐ２、Ｐ４のみが知る置換［π］ｉである。

【００７１】

以上の右辺の内容を入力とする出力［ｖｅｃ｛ｙ｝］ｍは、置換［π］４を用いたミニシャッフルを１回行った結果であり、［ｙπ４（１）］，・・・，［ｙπ４（ｍ）］のように表される。この出力中のｙπ４（ｊ）も、前述のように、ｙπ４（ｊ）＝ｙπ４（ｊ），１＋ｙπ４（ｊ），２＋ｙπ４（ｊ），３（ｊ＝１，・・・，ｍ）を満たす形で、秘密計算サーバＰｉに分散保持される。

【００７２】

次式［数１２］に上記したミニシャッフルの手順を示す。

【００７３】

【数１２】

1. P_4, P_1 and P_2 generate the random values $r_{j,1}$ and $r_{j,2}$.
   $r_{j,1} = H(vid_1||1||j,\ seed_2)$
   $r_{j,2} = H(vid_1||2||j,\ seed_2)$
2. We set $y_{\pi_4(j)} = y_{\pi_4(j),1} + y_{\pi_4(j),2} + y_{\pi_4(j),3}$.
   2-1.P_1 computes $y_{\pi_4(j),1}$ and $y_{\pi_4(j),2}$ as follows.
   $y_{\pi_4(j),1} = x_{\pi_4(j),1} - r_{j,1}$
   $y_{\pi_4(j),2} = x_{\pi_4(j),2} + r_{j,1} + r_{j,2}$
   2-2. P_2 computes $y_{\pi_4(j),2}$ and $y_{\pi_4(j),3}$ as follows.
   $y_{\pi_4(j),2} = x_{\pi_4(j),2} + r_{j,1} + r_{j,2}$
   $y_{\pi_4(j),3} = x_{\pi_4(j),3} - r_{j,2}$
   2-3. P_4 computes $y_{\pi_4(j),1} - y_{\pi_4(j),2}$ and $y_{\pi_4(j),2} - y_{\pi_4(j),3}$ as follows.
   $y_{\pi_4(j),1} - y_{\pi_4(j),2} = \left(x_{\pi_4(j),1} - x_{\pi_4(j),2}\right) - 2r_{j,1} - r_{j,2}$
   $y_{\pi_4(j),2} - y_{\pi_4(j),3} = \left(x_{\pi_4(j),2} - x_{\pi_4(j),3}\right) + r_{j,1} + 2r_{j,2}$
3. P_1 sends $\overrightarrow{m_{1,1}} = \overrightarrow{y_1}$ to P_3.
   P_2 sends $\overrightarrow{m_{3,2}} = \overrightarrow{y_3}$ to P_3.
   P_4 sends $\overrightarrow{m_4} = \overrightarrow{y_1} - \overrightarrow{y_3}$ to P_3.
   $\overrightarrow{y_i} = (y_{\pi_4(1),i}, \cdots, y_{\pi_4(m),i})$
4. P_3 checks whether $\overrightarrow{m_4} = \overrightarrow{m_{1,1}} - \overrightarrow{m_{3,2}}$ or not.
   If it holds, then P_3 outputs continue.
   1If it does not hold, then P_3 outputs abort.

【００７４】

（ステップ４－１）まず、秘密計算サーバＰ４、Ｐ１、Ｐ２は、受信ノードである秘密計算サーバＰ３が知らないシードｓｅｅｄ２を用いて、２つの乱数ｒｊ，１、ｒｊ，２を生成する。

【００７５】

（ステップ４－２－１～４－２－３）次に、秘密計算サーバＰ４、Ｐ１、Ｐ２は、互いに連携して、乱数ｒｊ，１、ｒｊ，２を用いて、ｙπ４（ｊ）＝ｙπ４（ｊ），１＋ｙπ４（ｊ），２＋ｙπ４（ｊ），３（ｍｏｄ　Ｒ）を満たすようなｙπ４（ｊ），１，ｙπ４（ｊ），２，ｙπ４（ｊ），３を計算する。具体的には、ｙπ４（ｊ），１，ｙπ４（ｊ），２，ｙπ４（ｊ），３は、以下のように計算される。

ｙπ４（ｊ），１＝ｘπ４（ｊ），１－ｒｊ，１
ｙπ４（ｊ），２＝ｘπ４（ｊ），２＋ｒｊ，１＋ｒｊ，２
ｙπ４（ｊ），３＝ｘπ４（ｊ），３－ｒｊ，２

【００７６】

さらに、秘密計算サーバＰ４は、［数１２］に示すように、ｙπ４（ｊ），１－ｙπ３（ｊ），２、ｙπ４（ｊ），２－ｙπ４（ｊ），３を計算する。

【００７７】

（ステップ４－３）次に、秘密計算サーバＰ４、Ｐ１、Ｐ２は、受信ノードである秘密計算サーバＰ３に対し、計算結果を送信する。具体的には、秘密計算サーバＰ１は、秘密計算サーバＰ３に対し、ｖｅｃ｛ｙ１｝をｖｅｃ｛ｍ１，１｝として送信する。秘密計算サーバＰ２は、秘密計算サーバＰ３に対し、ｖｅｃ｛ｙ３｝をｖｅｃ｛ｍ３，２｝として送信する。さらに、秘密計算サーバＰ４は、秘密計算サーバＰ３に対し、検証データとして、ｖｅｃ｛ｙ１｝－ｖｅｃ｛ｙ３｝をｖｅｃ｛ｍ４｝として送信する。なお、［数１２］に表したとおり、ｖｅｃ｛ｙｉ｝は、（［ｙπ４（１），ｉ］，・・・，［ｙπ４（ｍ），ｉ］）のようなｍ次元のベクトルである。

【００７８】

（ステップ４－４）次に、秘密計算サーバＰ３は、ｖｅｃ｛ｍ４｝＝ｖｅｃ｛ｍ１，１｝－ｖｅｃ｛ｍ３，２｝が成り立つか否かにより、不正検知を行う。ｖｅｃ｛ｍ４｝＝ｖｅｃ｛ｍ１，１｝－ｖｅｃ｛ｍ３，２｝が成り立つ場合、正しくミニシャッフルが行われたと判定し、処理を終了する。一方、ｖｅｃ｛ｍ４｝＝ｖｅｃ｛ｍ１，１｝－ｖｅｃ｛ｍ３，２｝が

成り立たない場合、秘密計算サーバＰ３は、不正なシャッフルが行われたと判定し、処理を中断する。

【００７９】

以上のミニシャッフルの結果、秘密計算サーバＰ１～Ｐ４は、次式［数１３］のように、ミニシャッフル後のｍ次元ベクトルを保持することになる。

【００８０】

【数１３】

$$
\begin{aligned}
&P_1: (\overrightarrow{y_1}, \overrightarrow{y_2}) = \left(\left(y_{\pi_4(1),1}, y_{\pi_4(1),2}\right), \cdots, \left(y_{\pi_4(m),1}, y_{\pi_4(m),2}\right) \right) = \left(\left[y_{\pi_4(1)}\right]_1, \cdots, \left[y_{\pi_4(m)}\right]_1 \right) \\
&P_2: (\overrightarrow{y_2}, \overrightarrow{y_3}) = \left(\left(y_{\pi_4(1),2}, y_{\pi_4(1),3}\right), \cdots, \left(y_{\pi_4(m),2}, y_{\pi_4(m),3}\right) \right) = \left(\left[y_{\pi_4(1)}\right]_2, \cdots, \left[y_{\pi_4(m)}\right]_2 \right) \\
&P_3: (-\overrightarrow{m_4} + \overrightarrow{m_{1,1}}, \overrightarrow{m_{1,1}}) \\
&\quad = (\overrightarrow{y_3}, \overrightarrow{y_1}) \\
&\quad = \left(\left(y_{\pi_4(1),3}, y_{\pi_4(1),1}\right), \cdots, \left(y_{\pi_4(m),3}, y_{\pi_4(m),1}\right) \right) = \left(\left[y_{\pi_4(1)}\right]_3, \cdots, \left[y_{\pi_4(m)}\right]_3 \right) \\
&P_4: (\overrightarrow{y_1} - \overrightarrow{y_2}, \overrightarrow{y_2} - \overrightarrow{y_3}) \\
&\quad = \left(\left(y_{\pi_4(1),1} - y_{\pi_4(1),2}, y_{\pi_4(1),2} - y_{\pi_4(1),3}\right), \cdots, \left(y_{\pi_4(m),1} - y_{\pi_4(m),2}, y_{\pi_4(m),2} - y_{\pi_4(m),3}\right) \right) \\
&\quad = \left(\left[y_{\pi_4(1)}\right]_4, \cdots, \left[y_{\pi_4(m)}\right]_4 \right)
\end{aligned}
$$

【００８１】

以上のようなミニシャッフルを行うことで、入力された［ｖｅｃ｛ｘ｝］ｍの置換［π］ｉを４回行うことが可能となる。また、上記各ラウンドで説明したとおり、［π］ｉによる置換は、受信ノードにその置換内容を知られない形態で行われるが、受信ノードは、不正を決定的検出できるようになっている。

【００８２】

上記した４ラウンドの置換は、下記［数１４］のように表すことができる。

【００８３】

【数１４】

$$
\begin{aligned}
&1.[\vec{w}]^m \leftarrow MiniShuffle(1, [\vec{v}]^m, [\pi_1]_1) \\
&2.[\vec{x}]^m \leftarrow MiniShuffle(2, [\vec{w}]^m, [\pi_2]_2) \\
&3.[\vec{y}]^m \leftarrow MiniShuffle(3, [\vec{x}]^m, [\pi_3]_3) \\
&4.[\vec{z}]^m \leftarrow MiniShuffle(4, [\vec{y}]^m, [\pi_4]_4)
\end{aligned}
$$

【００８４】

また、本実施形態によるシャッフルのコストは、環のサイズがκビットのｍ個の要素の通信を１ラウンドあたり３回行うので、計４ラウンドで１２κｍビットとなる。

【００８５】

［第２の実施形態］

上記した第１の実施形態では、４ラウンドを実行するものとして説明したが、計算の仕

方に工夫を加えることで、２ラウンドにて、第１の実施形態と同等のシャッフルを行うことができる。以下、２ラウンドにてシャッフルを行う第２の実施形態について説明する。第２の実施形態は、第１の実施形態と同様の構成で実施可能であるため、以下、その相違点を中心に説明する。

【００８６】

［ラウンド１　ｉ＝1、２］

図１１は、第２の実施形態のラウンド１（ｉ＝１、２）の動作を説明するための図である。図１１に示されたとおり、第２の実施形態においては、第１の実施形態のラウンド１とラウンド２の計算とデータの送信が並列に行われる。具体的には、秘密計算サーバＰ$_1$は、秘密計算サーバＰ$_4$に対し、ｖｅｃ｛ｙ$_1$｝－ｖｅｃ｛ｙ$_2$｝をｖｅｃ｛ｍ$_{1,1}$｝として送信する。秘密計算サーバＰ$_2$は、秘密計算サーバＰ$_4$に対し、ｖｅｃ｛ｙ$_2$｝－ｖｅｃ｛ｙ$_3$｝をｖｅｃ｛ｍ$_{2,2}$｝として送信する。さらに、秘密計算サーバＰ$_3$は、秘密計算サーバＰ$_4$に対し、検証データとして、ｖｅｃ｛ｙ$_1$｝－ｖｅｃ｛ｙ$_3$｝をｖｅｃ｛ｍ$_3$｝として送信する。これと並列に、秘密計算サーバＰ$_2$は、秘密計算サーバＰ$_1$に対し、ｖｅｃ｛ｙ$_2$｝をｖｅｃ｛ｍ$_{2,2}$｝として送信する。秘密計算サーバＰ$_3$は、秘密計算サーバＰ$_1$に対し、ｖｅｃ｛ｙ$_1$｝をｖｅｃ｛ｍ$_{1,3}$｝として送信する。さらに、秘密計算サーバＰ$_4$は、秘密計算サーバＰ$_1$に対し、検証データとして、ｖｅｃ｛ｙ$_1$｝－ｖｅｃ｛ｙ$_2$｝をｖｅｃ｛ｍ$_4$｝として送信する。

【００８７】

ここで、秘密計算サーバＰ$_2$は、ｖｅｃ｛ｍ$_{2,2}$｝とｖｅｃ｛ｍ$_{2,2}$｝を送信することになるが、［数３］のとおり、前者は、ｖｅｃ｛ｙ$_2$｝－ｖｅｃ｛ｙ$_3$｝であり、［数６］のとおり、後者は、ｖｅｃ｛ｙ$_2$｝であるので、並列して計算することができる。同様に、秘密計算サーバＰ$_3$は、ｖｅｃ｛ｍ$_3$｝とｖｅｃ｛ｍ$_{1,3}$｝を送信することになるが、［数３］のとおり、前者は、ｖｅｃ｛ｙ$_1$｝－ｖｅｃ｛ｙ$_2$｝であり、後者は、ｖｅｃ｛ｙ$_1$｝であるので、並列して計算することができる。

【００８８】

図１２は、第２の実施形態のラウンド２（ｉ＝３、４）の動作を説明するための図である。図１２に示されたとおり、第２の実施形態においては、第１の実施形態のラウンド３とラウンド４の計算とデータの送信が並列に行われる。具体的には、秘密計算サーバＰ$_1$は、秘密計算サーバＰ$_2$に対し、ｖｅｃ｛ｙ$_2$｝をｖｅｃ｛ｍ$_{2,1}$｝として送信する。秘密計算サーバＰ$_3$は、秘密計算サーバＰ$_2$に対し、ｖｅｃ｛ｙ$_3$｝をｖｅｃ｛ｍ$_{3,3}$｝として送信する。さらに、秘密計算サーバＰ$_4$は、秘密計算サーバＰ$_2$に対し、検証データとして、ｖｅｃ｛ｙ$_2$｝－ｖｅｃ｛ｙ$_3$｝をｖｅｃ｛ｍ$_4$｝として送信する。これと並列に、秘密計算サーバＰ$_1$は、秘密計算サーバＰ$_3$に対し、ｖｅｃ｛ｙ$_1$｝をｖｅｃ｛ｍ$_{1,1}$｝として送信する。秘密計算サーバＰ$_2$は、秘密計算サーバＰ$_3$に対し、ｖｅｃ｛ｙ$_3$｝をｖｅｃ｛ｍ$_{3,2}$｝として送信する。さらに、秘密計算サーバＰ$_4$は、秘密計算サーバＰ$_3$に対し、検証データとして、ｖｅｃ｛ｙ$_1$｝－ｖｅｃ｛ｙ$_3$｝をｖｅｃ｛ｍ$_4$｝として送信する。

【００８９】

ここで、秘密計算サーバＰ$_1$は、ｖｅｃ｛ｍ$_{2,1}$｝とｖｅｃ｛ｍ$_{1,1}$｝を送信することになるが、［数９］のとおり、前者は、ｖｅｃ｛ｙ$_2$｝であり、［数１２］のとおり、後者は、ｖｅｃ｛ｙ$_1$｝であるので、並列して計算することができる。同様に、秘密計算サーバＰ$_4$は、ｖｅｃ｛ｍ$_4$｝とｖｅｃ｛ｍ$_4$｝を送信することになるが、［数９］のとおり、前者は、ｖｅｃ｛ｙ$_2$｝－ｖｅｃ｛ｙ$_3$｝であり、後者は、ｖｅｃ｛ｙ$_1$｝－ｖｅｃ｛ｙ$_3$｝であるので、並列して計算することができる。もちろん、本実施形態においても、受信ノードが検証データを用いて、不正検知を行うことができる。

【００９０】

さらに、本実施形態では、前述のとおりラウンド数を２に削減することができる。本実施形態において、ハッシュ値の通信コストを無視すると、通信コストは、８κｍに削減することができる。

【００９１】
［第３の実施形態］
続いて、上記した第１の実施形態の検証ノード及び受信ノードにおける検証の方法に変更を加えた第３の実施形態について説明する。第３の実施形態は、第１の実施形態と同様の構成で実施可能であるため、以下、その相違点を中心に説明する。
【００９２】
第３の実施形態においては、ミニシャッフルの後半手順３．以降が次式［数１５］の手順に置き換えられる。
【００９３】
【数１５】

3. P_1 sends $\overrightarrow{m_{1,1}} = \overrightarrow{y_1} - \overrightarrow{y_2}$ to P_4.
P_2 sends $\overrightarrow{m_{2,2}} = \overrightarrow{y_2} - \overrightarrow{y_3}$ to P_4.
P_3 sends v_3 to P_4.
P_4 computes v'.
$\overrightarrow{y_i} = (y_{\pi_1(1),i}, \cdots, y_{\pi_1(m),i})$
$\overrightarrow{m_{1,1}} - \overrightarrow{m_{2,2}} = (m'_1, \ldots, m'_m), \overrightarrow{m_3} = \overrightarrow{y_1} - \overrightarrow{y_3} = (m_{3,1}, \ldots, m_{3,m})$としたとき,
$\alpha_j = H(vid_1||j,\ seed_3),\quad v' = \sum_{j=1}^{m} \alpha_j m'_j,\quad v_3 = \sum_{j=1}^{m} \alpha_j m_{3,j}$とする.
4. P_4 checks whether$v_3 = v'$ or not.
If it holds, then P_4 outputs *continue*.
If it does not hold, then P_4 outputs *abort*.

【００９４】
具体的には、秘密計算サーバＰ１は、秘密計算サーバＰ４に対し、ｖｅｃ｛ｙ１｝－ｖｅｃ｛ｙ２｝をｖｅｃ｛ｍ１，１｝として送信する。秘密計算サーバＰ２は、秘密計算サーバＰ４に対し、ｖｅｃ｛ｙ２｝－ｖｅｃ｛ｙ３｝をｖｅｃ｛ｍ２，２｝として送信する。ここまでの動作は第１の実施形態と同様である。第３の実施形態では、秘密計算サーバＰ３は、検証用データｖ３を計算し、秘密計算サーバＰ４に送信する（図１３参照）。一方、秘密計算サーバＰ４は、検証用データｖ’を計算する。
【００９５】
上記検証用データｖ３，ｖ’は次式［数１６］に規定される。
【００９６】
【数１６】

$$\overrightarrow{m_{1,1}} - \overrightarrow{m_{2,2}} = (m'_1, \ldots, m'_m), \overrightarrow{m_3} = \overrightarrow{y_1} - \overrightarrow{y_3} = (m_{3,1}, \ldots, m_{3,m})$$

としたとき，

$$\alpha_j = H(vid_1||j,\ seed_3), \qquad v' = \sum_{j=1}^{m} \alpha_j m'_j,\ v_3 = \sum_{j=1}^{m} \alpha_j m_{3,j}$$

【００９７】
そして、秘密計算サーバＰ４は、ｖ３＝ｖ’が成り立つか否かにより、不正検知を行う。ｖ３＝ｖ’が成り立つ場合、正しくミニシャッフルが行われたと判定し、秘密計算サーバＰ４は、次のラウンドに進む。一方、ｖ３＝ｖ’が成り立たない場合、秘密計算サーバＰ４は、不正なシャッフルが行われたと判定し、以降の処理を中断する。なお、上記検証にあたり、確率的にしか行えないので，ｖ’の生成及びｖ３の生成・送信はセキュリティパラメータκ回だけ繰り返すことが好ましい（但し、ｖ３の送信は１ラウンドにまとめて良い）。
【００９８】
ラウンド２以降も［数６］［数９］［数１２］の手順３．以降は、それぞれ次の［数１７］、［数１８］、［数１９］の手順に置き換えられる。

【００９９】
【数１７】

3. P_2 sends $\overrightarrow{m_{2,2}} = \overrightarrow{y_2}$ to P_1.
P_3 sends $\overrightarrow{m_{1,3}} = \overrightarrow{y_1}$ to P_1.
P_4 sends v_4 to P_1.
P_1 computes v'.
$\overrightarrow{y_i} = (y_{\pi_1(1),i}, \cdots, y_{\pi_1(m),i})$
$\overrightarrow{m_{2,2}} - \overrightarrow{m_{1,3}} = (m'_1, \ldots, m'_m)$, $\overrightarrow{m_4} = (m_{4,1}, \ldots, m_{4,m})$としたとき、
$\alpha_j = H(vid_1||j,\ seed_1)$, $v' = \sum_{j=1}^{m} \alpha_j m'_j$, $v_4 = \sum_{j=1}^{m} \alpha_j m_{4,j}$とする.

4. P_1 checks whether $v_4 = v'$ or not.
If it holds, then P_1 outputs *continue*.
If it does not hold, then P_1 outputs *abort*.

【０１００】
【数１８】

3. P_1 sends $\overrightarrow{m_{2,1}} = \overrightarrow{y_2}$ to P_2.
P_3 sends $\overrightarrow{m_{3,3}} = \overrightarrow{y_3}$ to P_2.
P_4 sends v_4 to P_2.
P_2 computes v'.
$\overrightarrow{y_i} = (y_{\pi_1(1),i}, \cdots, y_{\pi_1(m),i})$
$\overrightarrow{m_{2,1}} - \overrightarrow{m_{3,3}} = (m'_1, \ldots, m'_m)$, $\overrightarrow{m_4} = (m_{4,1}, \ldots, m_{4,m})$としたとき、
$\alpha_j = H(vid_1||j,\ seed_2)$, $v' = \sum_{j=1}^{m} \alpha_j m'_j$, $v_4 = \sum_{j=1}^{m} \alpha_j m_{4,j}$とする.

4. P_2 checks whether $v_4 = v'$ or not.
If it holds, then P_2 outputs *continue*.
If it does not hold, then P_2 outputs *abort*.

【０１０１】
【数１９】

3. P_1 sends $\overrightarrow{m_{1,1}} = \overrightarrow{y_1}$ to P_3.
P_2 sends $\overrightarrow{m_{3,2}} = \overrightarrow{y_3}$ to P_3.
P_4 sends v_4 to P_3.
P_3 computes v'.
$\overrightarrow{y_i} = (y_{\pi_1(1),i}, \cdots, y_{\pi_1(m),i})$
$\overrightarrow{m_{1,1}} - \overrightarrow{m_{3,2}} = (m'_1, \ldots, m'_m)$, $\overrightarrow{m_4} = (m_{4,1}, \ldots, m_{4,m})$としたとき、
$\alpha_j = H(vid_1||j,\ seed_3)$, $v' = \sum_{j=1}^{m} \alpha_j m'_j$, $v_4 = \sum_{j=1}^{m} \alpha_j m_{4,j}$とする.

4. P_3 checks whether $v_4 = v'$ or not.
If it holds, then P_3 outputs *continue*.
If it does not hold, then P_3 outputs *abort*.

【０１０２】
ラウンド２以降も同様であり、秘密計算サーバＰ４は、検証用データｖ４を計算し、受

信ノードとなる秘密計算サーバＰ$_i$に送信する（図１４～図１６参照）。一方、秘密計算サーバＰ$_4$は、検証用データｖ’を計算する。また、ラウンド２以降も、上記ｖ’及びv_4を用いた検証は、確率的にしか行えないので，ｖ’の生成，v_4の生成・送信はセキュリティパラメータκ回だけ繰り返すことが好ましい（但し、ｖ$_4$の送信は１ラウンドにまとめて良い）。

【０１０３】

上記検証用データｖ$_4$，ｖ’は次式［数２０］～［数２２］に規定される。

【０１０４】

【数２０】

$$\overrightarrow{m_{1,1}}-\overrightarrow{m_{2,2}}=(m_1^{'},\ldots,m_m^{'}),\overrightarrow{m_3}=\overrightarrow{y_1}-\overrightarrow{y_3}=(m_{3,1},\ldots,m_{3,m})$$

としたとき、

$$\alpha_j=H(vid_1||j,\ seed_3),\qquad v^{'}=\sum_{j=1}^{m}\alpha_j m_j^{'},\ v_3=\sum_{j=1}^{m}\alpha_j m_{3,j}$$

【０１０５】

【数２１】

$$\overrightarrow{m_{2,2}}-\overrightarrow{m_{1,3}}=(m_1^{'},\ldots,m_m^{'}),\overrightarrow{m_4}=(m_{4,1},\ldots,m_{4,m})$$

としたとき、

$$\alpha_j=H(vid_1||j,\ seed_1),\ v^{'}=\sum_{j=1}^{m}\alpha_j m_j^{'},\ v_4=\sum_{j=1}^{m}\alpha_j m_{4,j}$$

【０１０６】

【数２２】

$$\overrightarrow{m_{1,1}}-\overrightarrow{m_{2,2}}=(m_1^{'},\ldots,m_m^{'}),\overrightarrow{m_3}=\overrightarrow{y_1}-\overrightarrow{y_3}=(m_{3,1},\ldots,m_{3,m})$$

としたとき、

$$\alpha_j=H(vid_1||j,\ seed_3),\qquad v^{'}=\sum_{j=1}^{m}\alpha_j m_j^{'},\ v_3=\sum_{j=1}^{m}\alpha_j m_{3,j}$$

【０１０７】

以上、第３の実施形態によれば、ハッシュ関数を用いずに検証（不正検知）を行うことが可能となる。

【０１０８】

以上、本発明の各実施形態を説明したが、本発明は、上記した実施形態に限定されるものではなく、本発明の基本的技術的思想を逸脱しない範囲で、更なる変形・置換・調整を加えることができる。例えば、各図面に示したネットワーク構成、各要素の構成、データの表現形態は、本発明の理解を助けるための一例であり、これらの図面に示した構成に限定されるものではない。

【０１０９】

また、上記した第１から第３の実施形態に示した手順は、秘密計算ノード乃至秘密計算サーバＰｉとして機能するコンピュータ（図１７の９０００）に、これらの装置としての機能を実現させるプログラムにより実現可能である。このようなコンピュータは、図１７のＣＰＵ（Ｃｅｎｔｒａｌ　Ｐｒｏｃｅｓｓｉｎｇ　Ｕｎｉｔ）９０１０、通信インターフェース９０２０、メモリ９０３０、補助記憶装置９０４０を備える構成に例示される。すなわち、図１７のＣＰＵ９０１０にて、乱数生成プログラムや置換処理プログラムを実行し、その補助記憶装置９０４０等に保持された各計算パラメーターの更新処理を実施させればよい。

【０１１０】

即ち、上記した第１から第３の実施形態に示した秘密計算サーバＰｉの各部（処理手段、機能）は、これらの装置に搭載されたプロセッサに、そのハードウェアを用いて、上記した各処理を実行させるコンピュータプログラムにより実現することができる。

【０１１１】
最後に、本発明の好ましい形態を要約する。
［第１の形態］
（上記第１の視点によるシャッフルシステム参照）
［第２の形態］
上記したシャッフルシステムにおいて、前記受信ノードは、前記検証ノードから受信したデータを用いた前記ミニシャッフルの正当性を確認した結果、前記ミニシャッフルの正当性を確認できない場合、処理を中止する構成を採ることができる。
［第３の形態］
上記したシャッフルシステムにおいて、
前記４台の秘密計算ノードは、
３つのシード（$seed_1$，$seed_2$，$seed_4$）と秘密情報ｘのシェア（x_1，x_2）とを秘密分散して保持する第１の秘密計算ノードと、
３つのシード（$seed_2$，$seed_3$，$seed_4$）と秘密情報ｘのシェア（x_2，x_3）とを秘密分散して保持する第２の秘密計算ノードと、
３つのシード（$seed_3$，$seed_1$，$seed_4$）と秘密情報ｘのシェア（x_3，x_1）とを秘密分散して保持する第３の秘密計算ノードと、
３つのシード（$seed_1$，$seed_2$，$seed_3$）と秘密情報ｘのシェア（x_1-x_2，x_2-x_3）とを秘密分散して保持する第４の秘密計算ノードと、により構成され、
前記再分散ノード及び前記検証ノードは、前記受信ノードの知らないシードを用いて、２つの乱数を生成し、
前記再分散ノードは、前記ミニシャッフルの結果として、前記シェアに前記乱数の組み合わせを適用した結果を前記受信ノードに送信し、
前記検証ノードは、前記２台の再分散ノードが前記受信ノードに送信した結果の和又は差を計算して、前記受信ノードに送信する構成を採ることができる。
［第４の形態］
上記したシャッフルシステムにおいて、
前記第４の秘密計算ノードと、第１の秘密計算ノードとを受信ノードとするミニシャッフルと、を並列に実行し、
前記第２の秘密計算ノードと、第３の秘密計算ノードとを受信ノードとするミニシャッフルと、を並列に実行する構成を採ることができる。
［第５の形態］
（上記第２の視点によるシャッフル方法参照）
［第６の形態］
（上記第３の視点によるプログラム参照）
なお、上記第５～第６の形態は、第１の形態と同様に、第２～第４の形態に展開することが可能である。
【０１１２】
なお、上記の特許文献および非特許文献の各開示は、本書に引用をもって繰り込み記載されているものとし、必要に応じて本発明の基礎ないし一部として用いることが出来るものとする。本発明の全開示（請求の範囲を含む）の枠内において、さらにその基本的技術思想に基づいて、実施形態ないし実施例の変更・調整が可能である。また、本発明の開示の枠内において種々の開示要素（各請求項の各要素、各実施形態ないし実施例の各要素、各図面の各要素等を含む）の多様な組み合わせ、ないし選択（部分的削除を含む）が可能である。すなわち、本発明は、請求の範囲を含む全開示、技術的思想にしたがって当業者であればなし得るであろう各種変形、修正を含むことは勿論である。特に、本書に記載した数値範囲については、当該範囲内に含まれる任意の数値ないし小範囲が、別段の記載のない場合でも具体的に記載されているものと解釈されるべきである。さらに、上記引用した文献の各開示事項は、必要に応じ、本発明の趣旨に則り、本発明の開示の一部として、その一部又は全部を、本書の記載事項と組み合わせて用いることも、本願の開示事項に含

まれるものと、みなされる。

【産業上の利用可能性】

【０１１３】

本発明は、シャッフルシステムのほか、そのシャッフル機能を用いたソート処理を行う秘密分散システムに適用することができる。

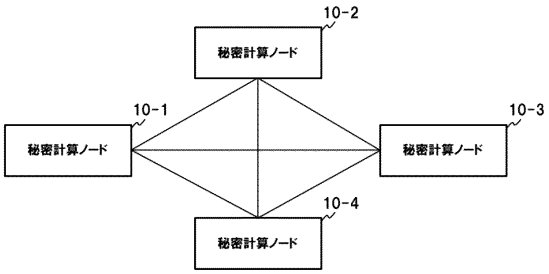
【符号の説明】

【０１１４】

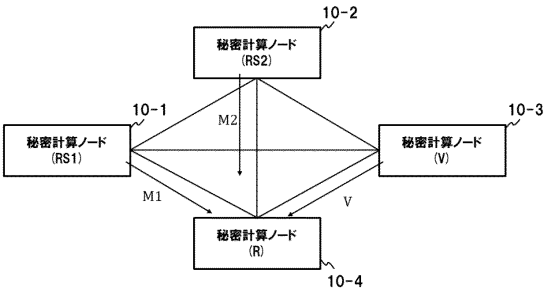
１０－１～１０－４　秘密計算ノード
１０１　置換生成部
１０２　置換適用部
１０３　算術演算部
１０４　不正検知部
１０５　乱数計算部
１０６　ハッシュ値計算部
１０７　シード記憶部
１０８　シェア値記憶部
９０００　コンピュータ
９０１０　ＣＰＵ
９０２０　通信インターフェース
９０３０　メモリ
９０４０　補助記憶装置
Ｐ$_1$～Ｐ$_4$　秘密計算サーバ

【図面】

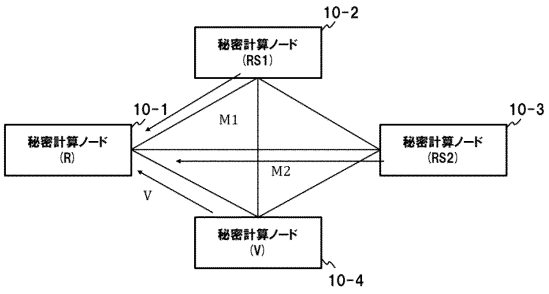
【図１】



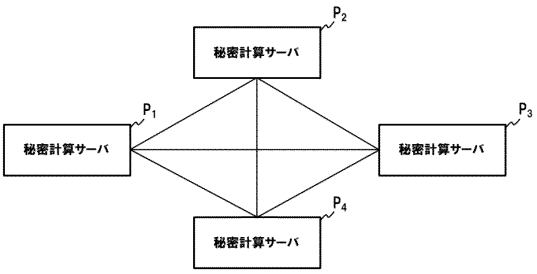
【図２】



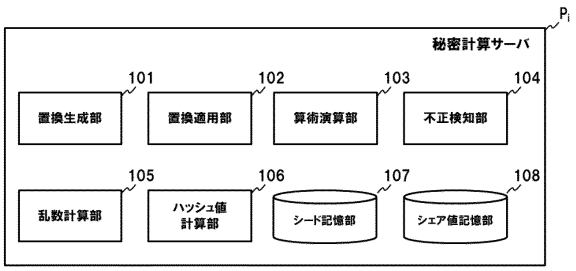
【図３】



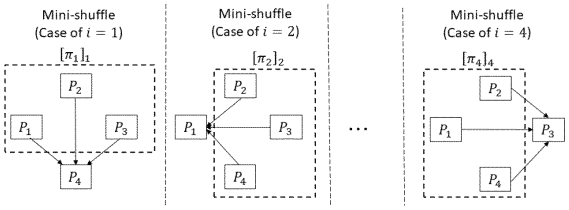
【図４】



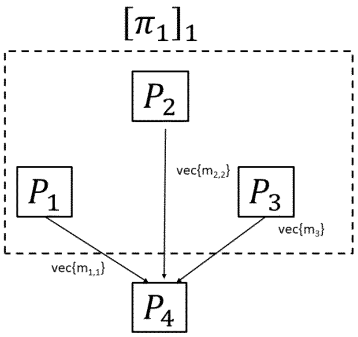
【図５】



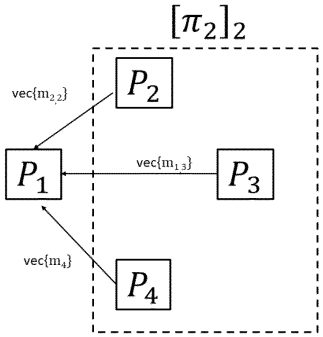
【図６】



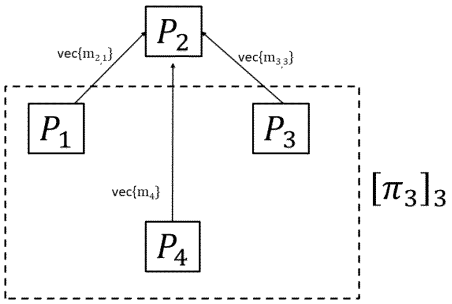
【図７】



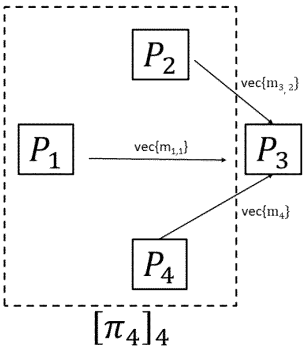
【図８】



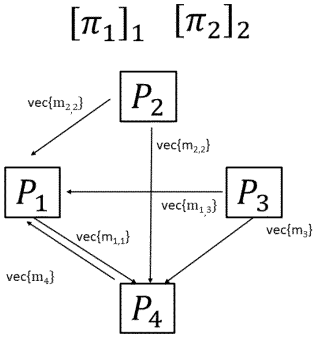
【図９】



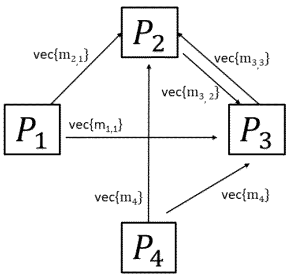
【図１０】



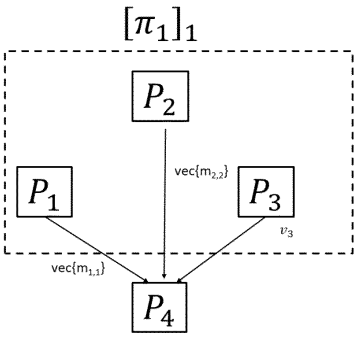
【図１１】



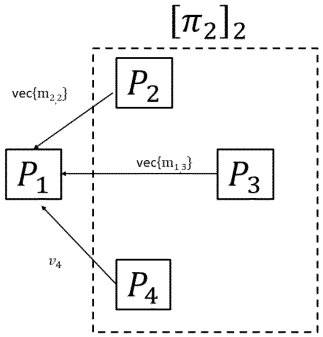
【図１２】



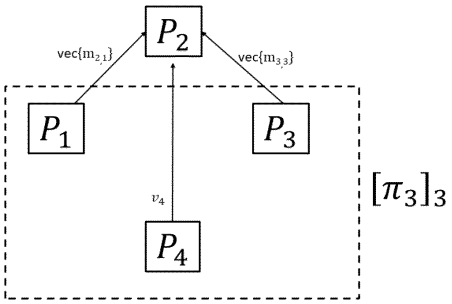
【図１３】



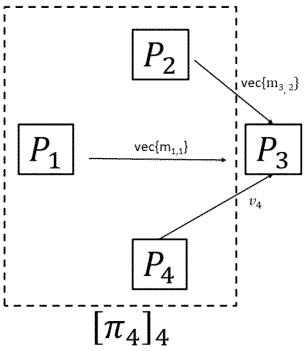
【図１４】



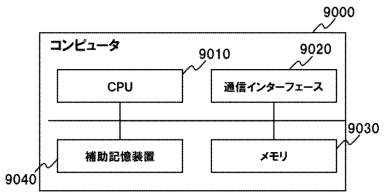
【図１５】



【図１６】



【図１７】



フロントページの続き

(56)参考文献 特開２０１７－１２９９１３（ＪＰ，Ａ）
国際公開第２０１２／０４６６９２（WO，Ａ１）
米国特許出願公開第２００３／００４６５４７（ＵＳ，Ａ１）
国際公開第２０１５／１０７９５２（WO，Ａ１）
辻下 健太郎 ほか，パスワード付秘密分散法の秘匿検索への応用，電子情報通信学会技術研究報告，日本，一般社団法人電子情報通信学会，2017年05月18日，Vol. 117 No. 55，p. 99-106
千田 浩司 ほか，効率的な３パーティ秘匿関数計算の提案とその運用モデルの考察，情報処理学会研究報告 コンピュータセキュリティ（ＣＳＥＣ），日本，社団法人情報処理学会，2010年04月15日，p. 1-7

(58)調査した分野 (Int.Cl.，ＤＢ名)
Ｇ０９Ｃ　１／００
Ｈ０４Ｌ　９／３２
ＪＳＴＰｌｕｓ／ＪＭＥＤＰｌｕｓ／ＪＳＴ７５８０（ＪＤｒｅａｍＩＩＩ）
ＩＥＥＥ Ｘｐｌｏｒｅ